2009-10-01

Context Aware Smart Classroom for Real Time Configuration of Computer Rooms

Paula Kelly
Technological University Dublin, paula.kelly@tudublin.ie

Peter Daly
Technological University Dublin, peter.daly@student.dit.ie

Ciaran O'Driscoll
Technological University Dublin, Ciaran.odriscoll@tudublin.ie

Recommended Citation

Kelly, P., Daly, P., & O'Driscoll, C. (2009). Context aware smart classroom for real time configuration of computer rooms. *Ninth IT & T Conference*, Technological University Dublin, Dublin, Ireland, 22nd.-23rd October. doi:10.21427/b6rh-qs98

Context Aware Smart Classroom for Real Time Configuration of Computer Rooms

Paula Kelly , Peter Daly , Ciaran O'Driscoll

Dublin Institute of Technology,
School of Electronic and Communications Engineering, Kevin Street
paula.kelly@dit.ie, peter.daly@student.dit.ie, ciaran.odriscoll@dit.ie

Abstract

The Context Aware Smart Classroom (CASC) is a classroom that responds to lecturers and student groups based on preset policies and the lecture timetables. CASC has been enhanced in two ways: initially to support the real-time software configuration of computers as required by specific laboratory activities; secondly to improve the decision making using knowledge engineering techniques. This paper outlines the design, implementation and evaluation of an enhanced system, CASC for Software Configuration (CASC–SC). Context aware environments respond in a pseudo-intelligent manner depending on the identity of occupants, particular location, desired activity and specific time. With the pervasive nature of personal mobile devices it is now possible to investigate development of low-cost location and identification systems that support development of a smart classroom

Keywords: Smart Classroom, Context, Context-Aware, Knowledge Engineering, Inference Engine

1 Introduction

The diversity of mature mobile personal electronic communication devices from mobile phones and PDAs to laptops presents the opportunity to develop truly Ubiquitous Computing [1] environments that can respond intelligently to occupants. In particular the use of such personal devices, supported by existing IT infrastructures, provides the possibility of developing cost effective Context Aware systems [2] for use in academia to enhance student learning experiences.

The Context Aware Smart Classroom to support real-time Software Configuration, CASC-SC, is designed to enable classrooms to make software configuration decisions for classroom PCs. Decisions are based on specific situational information such as location, identity of students and lecturers within the space, classroom timetables, and preset rules and policies. CASC–SC uses a rule based expert system to manage the reaction to changes in the environment according to rules in the system. The original smart classroom CASC [3] was developed to provide, real-time, context aware decisions, based on: information collected from environment sensors; policies; and rules of the smart classroom, in order to disseminate lecture material over WLAN, LAN or email during a class period.

CASC–SC is an enhanced version of CASC [3] and is focussed on in this work. CASC–SC provides additional functionality to support software configuration of computer rooms to ensure that only specific software is available during a class session as specified by preset policies. An additional enhancement provided by CASC-SC is the use of a Knowledge Engineering technique, in the form of an expert system based inference engine, to make appropriate decisions. The original version of CASC used nested `if-then-else` structures to parse the original rule set and policies. The inclusion of an inference engine will permit more complex rules to be used with the context data collected by the system.

2. Context Aware Smart Classroom

2.1 Context-aware

"*Context-aware computing is a mobile computing paradigm in which applications can discover and take advantage of contextual information (such as user location, time of day, nearby people and devices and user activity)*" [4]. This concept has been around for over a decade and it is only the recent availability of suitable portable computing and wireless network resources that make it possible to implement such systems.

The term context is used to describe real world situations, and everything is said to happen in a certain context. This makes it difficult to define context in a precise manner for many different situations. In computing the term "context-aware" was introduced in [5] and was applied to location information that could enable software to adapt according to its location of use, the identities of nearby people and objects, and changes to those objects over time

2.2 Context

Location is an essential element in defining context but it is by no means the only aspect that needs to be considered. Context in computing terms involves a number of different aspects. In [6] a definition for context with 3 elements is presented:

1. *Computing context, made up of nearby computing resources, communications and communications bandwidth.*
2. *User context, such as the user's profile, location, people nearby and even the social situation.*
3. *Physical context, such as lighting noise levels, traffic conditions and temperature.*

To more completely define context for computing, time was proposed as a fourth element in [4]:

4. *Time context, where user and physical contexts can be logged to provide a context history that can be useful in certain applications.*

These four particular aspects provide sufficient definition of context for the design and development of the context aware smart classroom.

Awareness of the context of the environment and the ability to react to changing context permits the development of pseudo intelligent or smart environments that can make autonomous decisions without the need to refer to users.

2.3 Smart Environments

Smart environments are an extension of the ubiquitous computing paradigm. One of the core concepts in ubiquitous computing is the ability of technology to disappear and become invisible to users [7, 8]. In the ubiquitous computing paradigm, Weiser [1] states, if a computer "*knows merely what room it is in, it can adapt its behaviour in significant ways without requiring even a hint of artificial intelligence*". While this is certainly the case, the addition of artificial intelligence techniques extends the potential range of behaviour and supports independent reaction.

Smart environments display a degree of autonomy, can adapt to changing situations and can communicate with users [9]. The provision of intelligent automation enhances ubiquitous computing environments and provides the opportunity for additional features such as detection of anomalous behaviour. Devices can easily be controlled using existing communications infrastructures based on sensor information collected and in particular predictive decision making can be included in the capabilities of the smart environment [10]. These capabilities allow an environment to exhibit pseudo-intelligent behaviour and so be considered as a smart environment.

2.4 Smart Classrooms

The development of "*applications are of course the whole point of ubiquitous computing*" [11] similarly in developing smart environments an experimental methodology is used as identified in [12]. This approach has lead to the development of a number of smart classrooms such as classroom 2000 [13] and eClass [14, 15] that were intended to reduce the workload of students. These systems automatically capture the lecture and make the material available on the web and this permits students to become more actively involved in the learning process during the class. The classroom 2000 and eClass research has targeted the capture and delivery of lectures using cameras and audio recording and supporting software infrastructure [12] to prepare notes for dissemination via the web.

CASC [3] was developed to disseminate lecture material used during a class period to students who had opted to participate. As part of the registration process for CASC, students provided a set of preferences for modes of receiving material such as Bluetooth, WLAN, or email etc. Details of personal Bluetooth enabled devices were also required as these were required by the system for location identification.

A key limitation of CASC was the use of nested `if-then-else` structures to make decisions. A more complex decision making approach using knowledge engineering was identified as a requirement for developing a campus wide system that could support more complex rule sets.

2.5 Knowledge Engineering

Knowledge engineering is the discipline that involves gathering and integrating knowledge into computer systems in order to solve complex problems normally requiring a high level of human expertise [16]. It is used in many computer science domains including expert systems which are a branch of Artificial Intelligence (AI) [17] that attempt to produce a solution to the level of a human expert in a specific problem domain by using the specialized knowledge that a human expert possesses. Knowledge is represented in two parts, facts and rules. Facts are data and the rules determine what the facts mean, e.g. consider a doctor using an expert system to choose the correct diagnosis based on a number of symptoms. Based on the facts that a patient has presented with a runny nose and a headache but no fever, an appropriate rule in the system might determine that the patient has a cold.

Expert systems attempt to reproduce the performance of human experts for a specific domain by creating a knowledge base of inference rules for that domain using some knowledge representation formalism and populating this knowledge base with information gathered from domain experts. Each inference rule is entered separately and an inference engine is used to infer information or take action based on the interaction of facts and the inference rules in the knowledge base.

2.6 Expert Systems

Expert systems are implemented using rule based languages rather than conventional procedural programming. In rule-based languages programs are composed by a set of inference rules. Each inference rule is composed of two parts, respectively called *left hand side (lhs)* and *right hand side (rhs)*. The program executes on the content of a specialized memory, called *working memory*. The working memory always contains a collection of records. The effect of a computation is the successive application of the inference rules in the program to the content of the working memory. The effect of applying an inference rule to the working memory is either the removal of a record or the introduction of a new record. The *rhs* of the inference rule contains the action used to modify the contents of the working memory. The *lhs* of each inference rule represents the conditions that have to be met for the rule to be applicable. Examples of rule based languages include CLIPS, Prolog and Jess (Java Expert System Shell).

2.6.1 The Rete algorithm

Expert systems with even moderately sized knowledge bases would perform too slowly if each rule had to be checked against known facts. The Rete algorithm [18] provides a more efficient implementation for an expert system . A Rete-based expert system builds a network of nodes, where each node (except the root) corresponds to a pattern occurring in the left-hand-side (the condition part) of a rule. The path from the root node to a leaf node defines a complete left-hand-side rule. Each node has a memory of facts which satisfy that pattern. As new facts are asserted or modified, they propagate along the network, causing nodes to be annotated when that fact matches the pattern. When a fact or combination of facts causes all of the patterns for a given rule to be satisfied, a leaf node is reached and the corresponding rule is triggered.

The Rete algorithm is designed to sacrifice memory for increased speed. In most cases, the speed increase over naïve implementations is several orders of magnitude (because Rete performance is theoretically independent of the number of rules in the system). In very large expert systems, however, the original Rete algorithm tends to run into memory consumption problems. Rete has become the basis for many popular expert system shells, including CLIPS and Jess.

2.6.2 Jess Rules Engine

Jess (Java Expert System Shell) [19] is a rules engine and scripting environment written in the Java programming language. The Jess language is derived from CLIPS in its syntax and uses the Rete algorithm for its pattern matching algorithm which makes Jess much faster than a simple set of cascading if-the statements adopted in conventional programming languages [19].

A Jess rule is something like an `if-then-else` statement in a procedural language, but it is not used in a procedural way. While `if-then-else` statements are executed at a specific time and in a specific order, according to how the programmer writes them, Jess rules are executed whenever their `if` parts (their *LHS*s) are satisfied, given only that the rule engine is running. It's architecture can be many orders of magnitude faster than an equivalent set of traditional `if-then-else` statements [19].

Jess provides several constructs to enable the construction of an expert system. Users can add their own functions to the language using native Jess code which makes Jess a powerful rule language facilitating users to execute their own Java classes in the Jess environment. The fact that Jess is written in Java allows its simple integration into Java applications.

3. CASC-SC System Design

Context Aware Smart Classroom for real-time Software configuration, CASC-SC, has been designed to enable the automatic software configuration of computers in a specific location for a particular class or laboratory session based on context data and a timetable schedule. In particular it is designed to ensure that only the programs required for the session are accessible to students. Preset policies determine the software requirements for a particular session. An inference engine, with appropriate rules, is used to ensure that the computers are configured in time for each session. The system is also designed to recognize different context events that occur, for example if a lecturer switches a session to a room other than that assigned in the timetable schedule, the system detects this (based on the location information for the class members) and the inference engine will automatically reconfigure the software for the PCs in the new room for that session. To help create a more efficient and secure environment the system will deny users access to PCs in a classroom if they are not scheduled to be in that classroom. Also, if a user tries to log into a PC and the system has not detected that the student is located within the room then the system will reject the login request as a security precaution.

3.1 CASC-SC Framework

CASC-SC is an enhancement of the CASC [3] prototype that disseminated material from a lecture to students. The framework architecture of CASC-SC is shown below in Figure 1. It is implemented as a client-server architecture to support distributed operation across a campus environment.

A presentation session is responsible for managing material shown during the class period. The smart classroom manager manages the adaptive behaviour and uses the inference engine to apply the rules for the system. Policies, set a priori, are retrieved from a policy manager database. Lecturers and students set policies related to their courses and specific personal devices that they will be using in the space. The policy manager identifies the appropriate activity for a classroom based on the timetable and identifies that the correct lecturer and students are present prior to commencing the session. The lecturer can set the note dissemination policy to determine the conditions required for students to receive different material developed in the session, such as to restrict dissemination of specific material to students present in the session. The context manager collects real-time data from environment sensors and a Bluetooth monitoring daemon running on the local client computer identifies individuals and communicates with their devices.

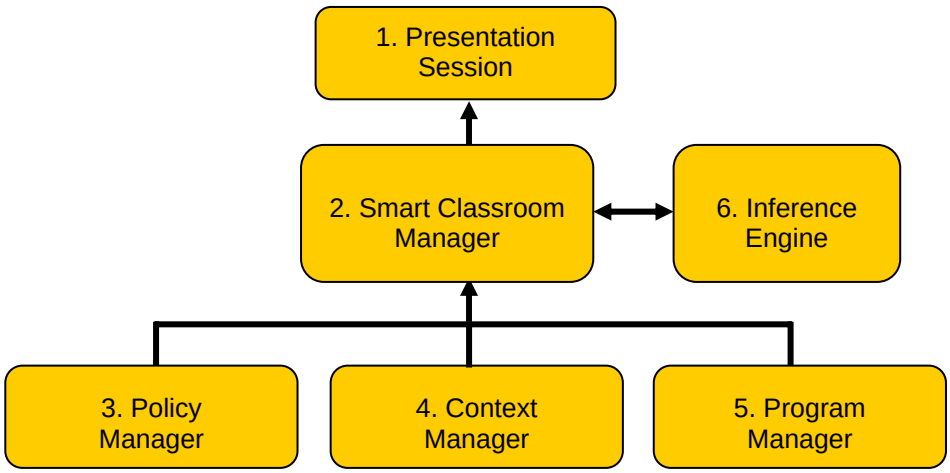

Figure 1: CASC-SC Framework

Using the design classifications identified by Baldauf et al. [2], the architecture of CASC-SC is a Context Server, deployed in a client-server model that implements a Logic Based Context Model. The context model is implemented using a Jess based inference engine based on facts and data from the database.

3.2 Inference Engine

The inference engine is an enhancement to the original CASC [3] prototype. A rules engine based on the Rete pattern matching algorithm [18] was designed to use information updated in database tables and make decisions based on the current system context and preset software configuration policies. The additional rules implemented for the prototype are:

- What software should be configured on the client PCs for a particular room based on those present in the room and the timetable schedule?
- Who should be allowed access to a classroom PC?

CASC-SC has been developed to support the automated software configuration of client PCs in individual smart rooms to suit the needs of the students and lecturer present within their environments and the timetable schedule. For instance if a programming lab is scheduled for a class the space should be able to configure itself to launch the appropriate software required for that lab, on the other hand if a lab is running a computer based exam the system should allow the lecturer to, for example, implement a policy to restrict access to the internet..

3.3 Program Manager

The program manager is a client that resides on a local PC in the smart classroom that is responsible for implementing a software configuration policy for the PCs in that room. This component denies users access to certain programs based on the classroom policy which is assigned by the decision server and stored in a central database. It also listens for login requests from users. Upon receipt of a login request it contacts the decision server to determine if the user is allowed access to the PC or not.

4 System Implementation

CASC-SC is a multi-threaded client-server architecture with a central server that manages the database tables and implements a Jess based rules engine to provide rule based decision-making functionality. Java was used as the core programming language for both client and server implementations with MySQL chosen as the database. The program scanner client machine and the decision server were both implemented on a Windows platform though any Linux operating system could equally be used. However the classroom scanner machine had to be Linux as CASC-SC uses a Linux API called BlueZ to manage the Bluetooth connections.

A component diagram showing the key components of CASC-SC is shown in Figure 2 below and the system was deployed as depicted in Figure 3.

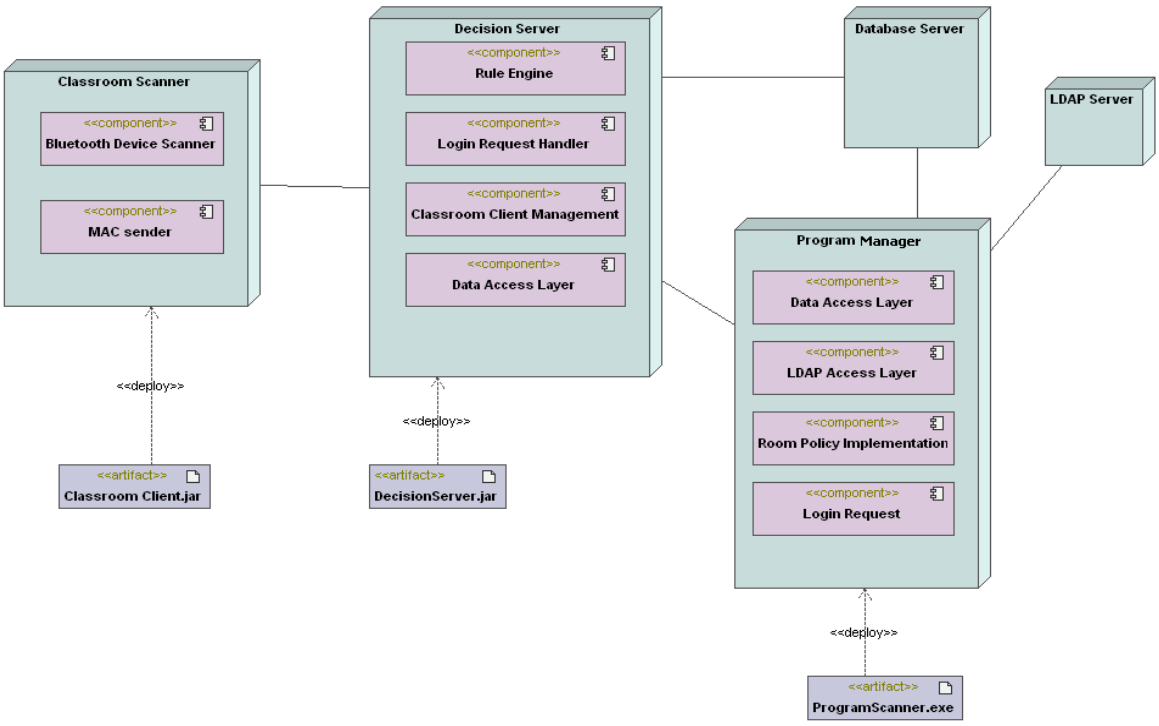

Figure 2: CASC-SC Component Diagram

4.1 Decision Server

The core and central server of the CASC-SC system is the Decision Server that provides the adaptive behaviour based on a set of system rules. It collects and updates the appropriate database tables with real-time context data from the classroom scanner component. It listens for login requests for users from the program scanner component. It uses the Jess rules engine to make decisions on whether a user is granted access to a PC in a particular classroom; and on the appropriate software configuration for user PCs in that classroom.

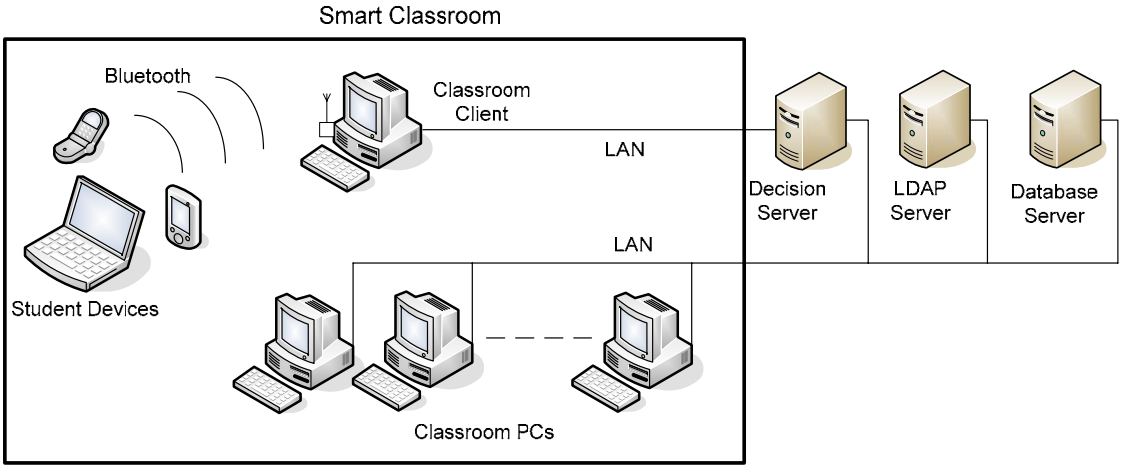

Figure 3: CASC-SC Deployment

4.2 Classroom Scanner

The classroom scanner component is a client program that resides on the Classroom Client PC in a smart classroom. It is responsible for supplying the decision server with the identities of the individuals in the space. This is implemented by scanning the space for Bluetooth devices and sending the MAC address data for any device found, as well as the room identity, to the decision server over a LAN connection.

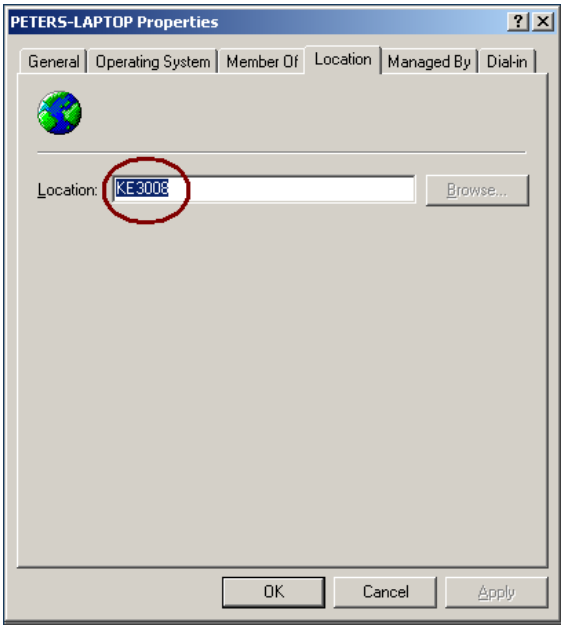

Figure 4. Room Location in the LDAP Server

4.4 Software Configuration of a Classroom PC

Each Classroom PC runs a local scanner client program which is responsible for monitoring user requests and implementing the policies on the local machine.

When the classroom PC is booted an application is started that first retrieves its room location from an LDAP server as shown in Figure 4. After the room location is retrieved the application calls the location verifier to process a login request with the decision server. Next the application "impersonates" a user with higher credentials on the PC and retrieves three policy lists from the database – the desktop list, the start-up list and the blocked list. It then writes the desktop shortcuts, as in Figure 4, from the desktop list as the impersonated user and launches the programs from the start up list. The application then retrieves the blocked list from the database and checks which processes are in the queue on the operating system. If a program from the blocked list is found it, the program scanner will use a system call to terminate the disallowed process. This process is then continued in a constant loop.

5. Evaluation

The system was evaluated over a 4 week period with two lecturer roles and several simulated student roles in order to determine the usability of the system. From this deployment, the system was evaluated from a technical perspective and recommendations developed for future enhancements. It was commented that the system removed tedious activities at the start of practical sessions and improved efficiency. It was also noted that it could potentially provide a management layer for the classroom that would abstract the lecturer from the technology.

5.1 Deployment

The CASC-SC system uses commonly available personal mobile devices supported by pervasive technologies such as Bluetooth, WLAN and LAN that make system deployment relatively simple and cheap to deploy in a real academic environment. The system is a client-server architecture which makes it easy to set up many smart rooms by adding the CASC-SC client with a Bluetooth dongle to existing theatre or lecture facilities. The CASC-SC system effectively manages the software configuration of the PCs within the smart classroom.

The rules engine, Jess, used in this system uses the Rete algorithm to process rules and is thus much faster than a simple set of cascading if-then-else statements adopted in conventional programming languages [20] and the original CASC implementation [3].

5.2 System Limitations

The Bluetooth sensor occasionally scanned extra devices that were not inside the room. This problem depended on the actual positioning of the Bluetooth monitoring sensor. If it was positioned near the smart classroom door Bluetooth devices outside the room could be scanned. In this case the location database would be updated with incorrect data indicating that a user was inside the smart space. However, this problem was alleviated by placing the sensor at an external wall, away from the door.

Another drawback when relying on Bluetooth technology to provide context data is that a user might not have their Bluetooth devices switched on resulting in the user not being recognised within the space. A better solution would be to use RFID as a means of identification. RFID tags respond much faster than Bluetooth devices and they can be integrated into a staff or student card.

Performance issues, in terms of system response time, were identified when using Bluetooth as a means of identifying people in the space. Testing showed that the Bluetooth Device Discovery Protocol requires 5 to 8 seconds to identify a device within the space.

A potential bottleneck in the system is the number of facts in the rules engine working memory that might have to be evaluated against the rules in the rules engine rules base. The solution employed in this implementation is to only load into the rules engine working memory with the timetable information for one hour at a time. In addition, as outlined in [19], the performance of a Rete-based

system depends not so much on the number of rules and facts but on the number of partial matches generated by the rules.

It might be argued that storing a person's current location in a database is a breach of his or her privacy rights. From a regulatory perspective users are required to opt-in, which meets EU requirements [21]. However in this implementation to provide security, the data is encrypted and only the decision server can decrypt it. In addition, the decision server generates a new random encryption key each time it runs to maintain security.

6. Conclusion

CASC-SC has successfully demonstrated that it provides enhanced functionality over the previous prototype smart classroom, CASC [3]. The purpose of developing both CASC and CASC-SC system has been to leverage existing technologies such as the personal devices of students and lecturers to enhance the students experience in the classroom. The smart classroom manager, in collaboration with the inference engine, was designed to automatically adapt to the behaviour of the room, based on the context, user policies and the core rules of the system. Bluetooth provided an acceptable solution to identifying users within the room although it occasionally identified users outside the room. The time for Bluetooth to identify each user raises concerns about the potential scalability of this identification technique. An alternative technique such as RFID tags in the student cards would probably improve performance and avoid mistaken identification. However this approach would require an RFID reader to be fitted at the doors of all smart classrooms and thus increase the cost and complexity of deployment. The CASC-SC system operated as an effective demonstration of the use of context awareness as a driver for creating a low-cost smart environment that can be developed using existing infrastructure and personal devices.

6.1 Future Work

With the inclusion of the inference engine, CASC-SC can be used to interpret more complex scenarios defined as additional rules. Such additional complexity will permit CASC-SC to be deployed as a campus wide solution and the basis of a Smart Campus service. Technical enhancements will be added to permit lecturers to set the software policies for each session or to use default session policies based on room capabilities. Location and latency issues related to Bluetooth will be addressed by using RFID as a means of identification of users. The distributed components of the system exchange data using low level communication protocols which limits the systems expandability capabilities. More study will be committed to using an XML based messaging service to support multiple categories of contextual data to be exchanged throughout the system. User acceptance and involvement with CASC-SC requires assessment over a longer period of time to establish the willingness of all stakeholders to engage with the system. A longitudinal research approach will be undertaken that will gather qualitative and quantitative research material through interviews and questionnaires.

References

[1] M. Weiser, "The computer for the 21st century," *Scientific American Vol. 265, No. 3, Sept. 1991, pp94-104, (Reprinted in Communications of ACM July 1993),* vol. 3, pp. 3-11.

[2] M. Baldauf, S. Dustdar, and F. Rosenberg, "A Survey on Context-Aware Systems," *Internation Journal of Ad Hoc and Ubiquitous Computing,* vol. 2, pp. 263 - 277, 2007.

[3] C. O'Driscoll, M. Mohan, F. Mtenzi, and B. Wu, "Deploying a Context Aware Smart Classroom," in *International Technology, Education and Development, InTED '08*, Valencia, Spain, 2008.

[4] G. Chen and D. Kotz, " A survey of Context-Aware Mobile Computing Research," Dartmouth College Computer Science TR2000-381, 2000.
[5] B. N. Schilit and M. M. Theimer, "Disseminating active map information to mobile hosts," *Network, IEEE,* vol. 8, p. 22, 1994.
[6] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Workshop on Mobile Computing Systems and Applications, Proceedings.*, 1994, p. 85.
[7] N. Streitz and P. Nixon, "Introduction to The Disappearing Computer," *Communications of the ACM,* vol. 48, pp. 32-35, 2005.
[8] D. M. Russell, N. A. Streitz, and T. Winograd, "Building disappearing computers," *Communications of the ACM,* vol. 48, pp. 42-48, 2005.
[9] S. K. Das, D. J. Cook, A. Battacharya, I. Heierman E. O., and A. T.-Y. L. Tze-Yun Lin, "The role of prediction algorithms in the MavHome smart home architecture," *Wireless Communications, IEEE,* vol. 9, p. 77, 2002.
[10] S. K. Das and D. J. Cook, "Designing and modeling smart environments," in *World of Wireless, Mobile and Multimedia Networks. WoWMoM* 2006, p. 5 pp.
[11] M. Weiser, "Some computer science issues in ubiquitous computing," *Communications of the ACM,* vol. 36, pp. 75-84, 1993.
[12] G. D. Abowd, "Software Engineering Issues for Ubiquitous Computing," in *Proceedings of the 21st International Conference on Software Engineering (ICSE '99)*, Los Angeles, CA, 1999.
[13] G. D. Abowd, "Classroom 2000: An Experiment with the Instrumentation of a Living educational Environment " *IBM Systems Journal,* vol. 38, pp. 508-530, 1999.
[14] J. Brotherton and G. D. Abowd, "eClass. Sixth Chapter " in *The Digital University: Building a Learning Community*, R. Hazemi, S. Hailes, and S. Wilbur, Eds.: Springer Verlag, 2002, p. 252.
[15] J. A. Brotherton and G. D. Abowd, "Lessons learned from eClass: Assessing automated capture and access in the classroom," *ACM Transactions Computer-Human Interaction.,* vol. 11, pp. 121-155, 2004.
[16] E. Feigenbaum, McCorduck, P., Forgy, C.L., The fifth generation (1st ed.),. Reading, MA. Addison Wesley, ISBN 9780201115192, OCLC 9324691, 1983.
[17] J. Giarratano, Riley, G., Expert Systems: Principles and Programming, Thomson Press, 2004.
[18] C. L. Forgy, On the efficient implementation of production systems. Ph.D. Thesis, Carnegie-Mellon University, 1979.
[19] S. J. Sandina Labratories. (1997, the Rule Engine for the Java Platform. Retrieved April 09, 2009, from Jess, the Rule Engine for the Java Platform: http://www.jessrules.com/jess/index.shtml.
[20] S. Yang, Zhang, J., Chen, O., A Jess enabled context elicitation system for providing context-aware Web services, Expert Systems with Applications, Volume 34, Issue4. 2008.
[21] "Data Protection Directive (95/46/ec) ", 1995.