Formal Specification of a Context-aware Whiteboard System in CCA

Najat Atbaiga
Sirte University
Sirte, Libya
Email: najatatbaiga@gmail.com

François Siewe
Software Technology Research Laboratory
De Montfort University, Leicester LE1 9BH, UK
Email: fsiewe@dmu.ac.uk

***Abstract*—A context-aware whiteboard system provides a number of services in a smart classroom including registering students as they enter the classroom; logging students and lecturers in to the blackboard virtual learning environment at the beginning of each lecture and logging them out at the end of the lecture. This system also notifies students of their absence to a lecture and maintains a list of attendance automatically. Using information from the timetable, it is aware of the lectures that are scheduled to take place in the classroom and the students that are allowed to attend these lectures. Finally, it allows students and lecturers to interact with teaching materials such as lecture slides and videos stored in the blackboard virtual learning environment. This paper proposes a formal specification of the white board system in the Calculus of Context-aware Ambients (CCA in short). This enables the formal analysis of the white board system using the execution environment of CCA. Some important properties of a classroom white board system have been validated as a proof of concept.**

Keywords*—Calculus of Context-aware Ambients; pervasive system; CCA; formal specification; context-aware whiteboard

I. Introduction

The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it. This was Mark Weiser's vision for the computers of the 21st century [7]. This vision led to a new paradigm for distributed systems coined ubiquitous computing, sometimes known as pervasive computing. In order to make the user and his tasks the central focus rather than computing devices and technical issues, appliances should vanish into the background [1] and sense the user context so as to provide due assistance to the user any time and anywhere. Such systems are also known as context aware systems.

Pervasive systems can be utilised in any domain including education, and can save time, acting as a fast and easy way for producing services. Interruptions can happen in a university lecture due to organisational processes of the university such as registering attendance of students and customising classrooms.

A context-aware whiteboard system provides a number of services in a smart classroom including registering students as they enter the classroom; logging students and lecturers in to the blackboard virtual learning environment at the beginning of each lecture and logging them out at the end of the lecture. This system also notifies students of their absence to a lecture and maintains a list of attendance automatically. Using information from the timetable, it is aware of the lectures that are scheduled to take place in the classroom and the students that are allowed to attend these lectures. Finally, it allows students and lecturers to interact with teaching materials such as lecture slides and videos stored in the blackboard virtual learning environment. This paper proposes a formal specification of the white board system in the Calculus of Context-aware Ambients (see Sect. IV). This enables the formal analysis of the white board system using the execution environment of CCA. Some important properties of a classroom white board system have been validated as a proof of concept (see Sect. V).

II. Overview of Context-aware Whiteboard

A context-aware whiteboard (CaWB) system is able to register students attendance automatically as they enter the classroom; it is assumed that students carry ID cards with them and that CaWB can detect the presence of such ID cards once the students enter the classroom. The CaWB also detects the presence of a lecturer in the same way, by sensing the presence of the lecturer ID card in the classroom. When the CaWB detects the presence of a student, it checks with the blackboard (BB) system if that individual is enrolled on the module scheduled to take place in the classroom at that specific time. If the student is enrolled on that module, the CaWB logs in the student to the BB system and adds the student ID to the attendance list. If the student is not enrolled on that module, CaWB will send a notification to the student immediately to let her know she is probably in the wrong classroom.

When a lecturer enters the classroom, the CaWB checks if she is scheduled to teach in that classroom at that specific time. If so, the lecturer is automatically logged in to the BB system, otherwise the CaWB notifies her of being in the wrong classroom. Once logged in to the BB system, the lecturer can access the teaching materials such as lecture slides, tutorials, audio contents and videos. At the end of a class, the lecturer can check the attendance list and if necessary contact the students that did not turn up.

This general architecture shows the general components of the system. The ID card should be carried by the user "lecturers and students". Sensor hardware senses the data from a surrounding environment and communicates with middle ware to transfer the data wirelessly using for example Bluetooth or IR signal. The middle ware communicates with the context server using a network based on TCP/IP as an example. The middle ware in this system would be the white board. The context server updates and synchronises information across the system. Fig. 1 depicts the general architecture of CaBW system.

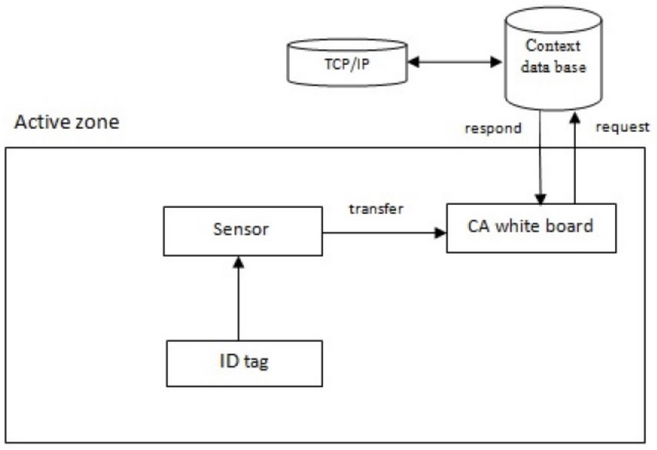


Fig. 1. Architecture of CaWB system

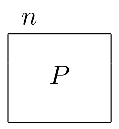
TABLE I. SYNTAX OF CCA

P, Q	$::=$		**Process**
		$\mathbf{0}$	inactivity
		$P \mid Q$	parallel composition
		$(\nu\ n)\ P$	name restriction
		$!P$	replication
		$n[P]$	ambient
		$\kappa?M.P$	context-guarded prefix
		$x \triangleright (y_1, \ldots, y_\ell).P$	process abstraction
		if $\kappa_1?M_1.P_1\ \ldots\ \kappa_\ell?M_\ell.P_\ell$ fi	selection
κ	$::=$		**Context Expressions**
		true	true
		$n = m$	name match
		$\bullet$	hole
		$\neg\kappa$	negation
		$\kappa_1 \mid \kappa_2$	parallel composition
		$\kappa_1 \wedge \kappa_2$	conjunction
		$n[\kappa]$	location context
		$\oplus\kappa$	spatial next modality
		✧κ	somewhere modality
M	$::=$		**Capabilities**
		del n	delete ambient n
		in n	move into ambient n
		out	move out of parent
		$\alpha\ x(z_1, \ldots, z_\ell)$	process abstraction call
		α recv$(y_1, \ldots, y_\ell)$	input
		α send$(z_1, \ldots, z_\ell)$	output
α	$::=$		**Locations**
		$\uparrow$	any parent
		$n\uparrow$	parent n
		$\downarrow$	any child
		$n\downarrow$	child n
		$::$	any sibling
		$n\ ::$	sibling n
		ϵ	locally

III. Overview of CCA

This section presents the syntax and the informal semantics of CCA. Table I depicts the syntax of CCA, based on three syntactic categories: processes (denoted by P or Q), capabilities (denoted by M) and context-expressions (denoted by κ). We assume a countably-infinite set of names, elements of which are written in lower-case letters, e.g. n, g, x and y.

Processes: The process $\mathbf{0}$, aka *inactivity process*, does nothing and terminates immediately. The process $P|Q$ denotes the concurrent execution of the processes P and Q. The process $(\nu\ n)\ P$ creates a new name n and the scope of that name is limited to the process P. The replication $!P$ denotes a process which can always create a new copy of P, i.e. $!P$ is equivalent to $P|!P$. Replication, first introduced in the π-Calculus can be used to implement both iteration and recursion. The process $n[P]$ denotes an ambient named n whose behaviours are described by the process P. The pair of square brackets '[' and ']' outlines the boundary of that ambient. An ambient $n[P]$ is represented graphically as follows:



A context expression specifies a property upon the state of the environment. A *context-guarded prefix* $\kappa?M.P$ is a process that waits until the environment satisfies the context expression κ, then performs the capability M and continues like the process P. The dot symbol '.' denotes the sequential composition of processes. We let $M.P$ denote the process true$?M.P$, where true is a context expression satisfied by all context. A process abstraction $x \triangleright (y_1, \ldots, y_\ell).P$ denotes the linking of the name x to the process P where $y_1, \ldots, y_\ell$ are the *formal parameters*. This linking is local to the ambient where the process abstraction is defined. So a name x can be linked to a specific process in one ambient and to a different process in another ambient. This locality property is interesting for context-awareness, especially when combined with ambient mobility. Hence an ambient calling a process abstraction may behave differently depending on its current location. A *selection* process if $\kappa_1?M_1.P_1\ \ldots\ \kappa_\ell?M_\ell.P_\ell$ fi waits until at least one of the context-expressions $(\kappa_i)_{1\leq i\leq \ell}$ holds; then proceeds non-deterministically like one of the processes $\kappa_j?M_j.P_j$ for which κ_j holds.

Capabilities: A call to a process abstraction named x is done by a capability of the form $\alpha\ x(z_1, \ldots, z_\ell)$ where α specifies the location where the process abstraction is defined and $z_1, \ldots, z_\ell$ are *actual parameters*. The location α can be '$\uparrow$' for any parent, '$n\uparrow$' for a specific parent n, '$\downarrow$' for any child, '$n\downarrow$' for a specific child n, '::' for any sibling, 'n ::' for a specific sibling n, or ϵ (empty string) for the calling ambient itself. A process call $\alpha\ x(z_1, \ldots, z_\ell)$ behaves like the process linked to x at location α, in which each actual parameter z_i, $i = 1, \ldots, \ell$ is substituted for each occurrence of the corresponding formal parameter. A process call can only take place if the corresponding process abstraction is *defined* at the specified location.

Ambients exchange messages using the output capability α send$(z_1, \ldots, z_\ell)$ to send a list of names $z_1, \ldots, z_\ell$ to a location α, and the input capability α recv$(y_1, \ldots, y_\ell)$ to receive a list of names from a location α. The mobility capabilities in and out are defined as follows. An ambient that performs the capability in n moves into the sibling ambient n. The capability out moves the ambient that performs it out of that ambient's parent. The capability del n deletes an ambient of the form $n[\mathbf{0}]$ situated at the same level as that capability, i.e. the process del $n.P \mid n[\mathbf{0}]$ reduces to P. The capability del acts as a garbage collector that deletes ambients which have completed their computations.

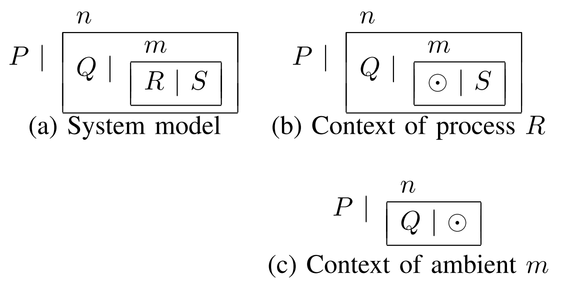

$P \mid n[Q \mid m[R \mid S]]$
(a) System model
$P \mid n[Q \mid m[\odot \mid S]]$
(b) Context of process R
$P \mid n[Q \mid \odot]$
(c) Context of ambient m

Fig. 2. Graphical illustration of the context of a process

Example 3.1:

- The process $n[\mathtt{in}\ m.\mathtt{out}.\mathbf{0}] \mid m[\mathtt{in}\ n.\mathtt{out}.\mathbf{0}]$ describes the behaviours of two sibling ambients n and m concurrently willing to move in and out of one another.
- The ambient $n[(\diamond\mathtt{at}(m))? :: \mathtt{send}(msg).\mathbf{0}]$ releases the message 'msg' only when at location m; where the context expression $\diamond\mathtt{at}(m)$ holds if n is a child ambient of the ambient m. The formal definition of the predicate '$\mathtt{at}$' is given in Example 3.2.

Context model: In CCA, a context is modelled as a process with a hole in it. The hole (denoted by $\odot$) in a context represents the position of the process that context is the context of. For example, suppose a system is modelled by the process $P \mid n[Q \mid m[R \mid S]]$. So, the context of the process R in that system is $P \mid n[Q \mid m[\odot \mid S]]$, and that of the ambient named m is $P \mid n[Q \mid \odot]$ as depicted graphically in Fig. 2. Thus the contexts of CCA processes are described by the grammar in Table II. A property of a context can be described by a formula called a *context expression* (CE in short).

TABLE II. SYNTAX OF CONTEXTS

$\mathcal{C}$	$::=$	$\mathbf{0} \mid \odot \mid n[\mathcal{C}] \mid `\mathcal{C}\|P' \mid (\nu\ n)\ \mathcal{C}$

Context expressions: The CE $\mathtt{true}$ always holds. A CE $n = m$ holds if the names n and m are lexically identical. The CE $\bullet$ holds solely for the hole context, i.e. the position of the process evaluating that context expression. Propositional operators such as negation ($\neg$) and conjunction ($\wedge$) expand their usual semantics to context expressions. A CE $\kappa_1|\kappa_2$ holds for a context if that context is a parallel composition of two contexts such that κ_1 holds for one and κ_2 holds for the other. A CE $n[\kappa]$ holds for a context if that context is an ambient named n such that κ holds inside that ambient. A CE $\oplus\kappa$ holds for a context if that context has a child context for which κ holds. A CE $\diamond\kappa$ holds for a context if there exists somewhere in that context a sub-context for which κ holds. The operator $\diamond$ is called *somewhere modality*, while $\oplus$ is aka *spatial next modality*.

Example 3.2: We now give some examples of predicates that can be used to specify common context properties such as the location of the user and whom the user is with. In these sample predicates we take the view that a process is evaluated by the immediate ambient λ say that contains it.

- $\mathtt{has}(n) = \oplus\ (\bullet \mid n[\mathtt{true}] \mid \mathtt{true})$
  holds if λ is top ambient and contains an ambient named n
- $\mathtt{at}(n) = n[\oplus(\bullet \mid \mathtt{true})] \mid \mathtt{true}$
  holds if λ is located at a top ambient named n
- $\mathtt{with}(n) = n[\mathtt{true}] \mid \oplus(\bullet \mid \mathtt{true})$
  holds if λ is (co-located) with an ambient named n at a top ambient.
- $\mathtt{at2}(n, m) = n[m[\mathtt{true}] \mid \mathtt{true}] \mid \mathtt{true}$
  holds if the ambient m is located in the ambient n.

IV. Formal Specification of CaWB System

In this section, the CaWB system is formally specified in CCA. It is assumed that the CaWB system is located in a classroom which acts as the active zone of the system, i.e. any ID card within the active zone can be detected by the CaWB, but ID cards outside the active zone cannot be detected by the CaWB system. Therefore, students outside the classroom are not detected by the CaWB, but are detected by the system once they enter the classroom; same for lecturers. The CaWB system is modelled in CCA using the following ambients:

- The classroom ambient *cRoom*, which represents the active zone of the system
- The blackboard ambient *BB*, which models the context server (see Fig. 1) and hosts the users login accounts
- The display system ambient *Display*, for powerpoint presentations and tutorials
- The audio-vision system ambients *AVS*, for playing audio contents and videos
- The attendance monitoring system *aList*, that manages students attendance
- The context-aware whiteboard ambient *CaWB*, that controls the whole system
- The lecturer ambient *Lect1*, representing a lecturer of ID number *Lect1*
- The student ambient *St1*, representing a student of ID number *St1*.

The overall model of the system is given in Fig. 3, where P_i is a process specifying the behaviour of the corresponding ambient that contains it.

The textual representation of the model in Fig. 3 is given in Eq. (1).

$$\left.\begin{array}{l} cRoom[\\ \quad CaWB[\\ \qquad P_1 \\ \qquad \mid BB[P_2] \\ \qquad \mid Display[P_3] \\ \qquad \mid AVS[P_4] \\ \qquad \mid aList[P_5] \\ \quad] \\ ] \\ \mid Lect1[P_6] \mid Lect2[P_6] \mid St1[P_7] \mid St2[P_7] \end{array}\right\} \quad (1)$$

The classroom ambient *cRoom* is the active zone of the CaWB system; this means that any object inside the *cRoom*

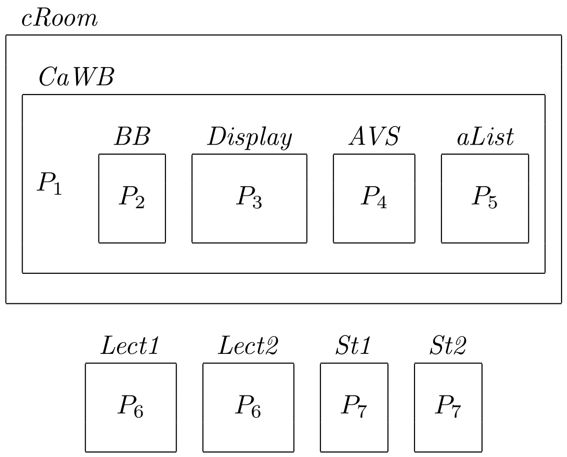


Fig. 3. The model of CaWB system in CCA

can be detected by the CaWB system. Initially, the classroom is empty; students and lecturers are waiting outside as depicted in Fig. 3 (graphically) and Eq. 1 (in the textual form). The behaviours $P_i, 1 \leq i \leq 7$ of the ambients are specified below.

A. The context-aware whiteboard ambient $CaWB$

This ambient continuously scans its active zone (i.e. the classroom) to detect the presence of students and lecturers in the classroom. Once a user is detected, the system automatically attempts to log the user in to the Blackboard (BB). This behaviour is specified in Eq. (2)

$$! :: \mathtt{recv}(src, req).BB\downarrow\mathtt{send}(src, req).\mathbf{0} \tag{2}$$

If the login is successful, a notification *login allowed* is sent to the user and that user's ID is added to the attendance list. If the login is denied, the user is notified and the attendance list is not updated. In addition to the login request, the CaWB ambient enables users to display the lecture slides and tutorials; to play audio and video contents using the AVS system. This behaviour is specified in Eq. (3). Therefore the whole

$$\left.\begin{array}{l}
!\downarrow\mathtt{recv}(dest, req, reply).\mathtt{if} \\
\quad (req = slides \vee req = tutorials)?Display\downarrow\mathtt{send}(req, reply).\mathbf{0} \\
\quad (req = video \vee req = audio)?AVS\downarrow\mathtt{send}(req, reply).\mathbf{0} \\
\quad (reply = allowed \wedge \text{✧}\mathtt{at}\mathit{2}(aList, dest))?dest :: \mathtt{send}(req, reply).\mathbf{0} \\
\quad (reply = allowed \wedge \neg\text{✧}\mathtt{at}\mathit{2}(aList, dest))?aList\downarrow\mathtt{send}(dest). \\
\qquad\qquad\qquad dest :: \mathtt{send}(req, reply).\mathbf{0} \\
\quad \mathtt{fi}
\end{array}\right\} \tag{3}$$

behaviour P_1 of the CaWB ambient is the parallel composition of the process in Eq. (2) and the process in Eq. (3), as depicted in Eq. (4).

$$P_1 \ = \ Eq.(2) \mid Eq.(3) \tag{4}$$

B. The blackboard ambient BB

The BB ambient maintains the database of students enrolled on a module and the lecturers involved in the teaching of that module. This database is simply modelled as a process of the form $m_1[\mathbf{0}] \mid \ldots \mid m_\ell[\mathbf{0}]$, where m_i, $1 \leq i \leq \ell$ are distinct user ID numbers. When The CaWB emits a login request for a user say x, the BB ambient checks if the user x is enrolled on the module (using the context expression $\text{✧}has(x)$, as shown in Eq. (5)), in which case the login request is successful; otherwise the login request is denied. The BB ambient also handles requests to access the teaching materials such as lecture slides (.ppt), tutorials (.pdf), audio (.wav) and video contents (.wmv). Thus, the behaviour P_2 of the BB ambient is given in Eq. (5). Note that for illustration, of the

$$P_2 = \left(\begin{array}{l}
!\uparrow\mathtt{recv}(src, req).\mathtt{if} \\
\ (req = login \wedge \text{✧}\mathtt{has}(src))?\uparrow\mathtt{send}(src, req, allowed).\mathbf{0} \\
\ (req = login \wedge \neg\text{✧}\mathtt{has}(src))?\uparrow\mathtt{send}(src, req, denied).\mathbf{0} \\
\ (req = slides)?\uparrow\mathtt{send}(src, req, slides\_ppt).\mathbf{0} \\
\ (req = tutorials)?\uparrow\mathtt{send}(src, req, tutorial\_pdf).\mathbf{0} \\
\ (req = video)?\uparrow\mathtt{send}(src, req, mm01\_wmv).\mathbf{0} \\
\ (req = audio)?\uparrow\mathtt{send}(src, req, mm01\_wav).\mathbf{0} \\
\mathtt{fi} \\
\mid Lect1[\mathbf{0}] \mid St1[\mathbf{0}]
\end{array}\right) \tag{5}$$

four users waiting outside the classroom in Fig. 3 only $Lect1$ and $St1$ are enrolled on the module in the blackboard BB as shown in Eq. (5).

C. The lecturer ambient Lect

It is assumed that a lecturer carries a smart badge (e.g. an RFID badge) that can interact wirelessly with the CaWB system. When a lecturer enters a classroom, the CaWB reads the ID number of the lecturer and automatically attempts to log the lecturer in to the blackboard system BB. If the login is successful, the lecturer is then allowed to request access to teaching materials such as slides, tutorials, audio and video contents. If in the contrary the login request is unsuccessful, the lecturer will be denied access to these resources and certainly will leave the classroom. The general behaviour P_6 of a lecturer is depicted in Eq. (6).

$$P_6 = \left(\begin{array}{l}
\mathtt{in}\ cRoom.CaWB :: \mathtt{send}(lecturer, login). \\
CaWB :: recv(req, reply).\mathtt{if} \\
(reply = allowed)?CaWB :: \mathtt{send}(lecturer, slides). \\
\quad CaWB :: \mathtt{send}(lecturer, tutorials). \\
\quad CaWB :: \mathtt{send}(lecturer, audio).\mathtt{out}.\mathbf{0} \\
(reply = denied)?\mathtt{out}.\mathbf{0} \\
\mathtt{fi}
\end{array}\right) \tag{6}$$

D. The student ambient St

Similarly to a lecturer, it is assumed that a student carries a smart badge (e.g. an RFID badge) that can interact wirelessly with the CaWB system. When a student enters a classroom, the CaWB reads the ID number of the student and automatically attempts to log the student in to the blackboard system BB. If the login is successful, the student's ID is added to the attendance list; otherwise the student is notified accordingly and eventually leaves the classroom. The general behaviour of a student is given in Eq. (7).

$$P_7 = \left(\begin{array}{l}
\mathtt{in}\ cRoom.CaWB :: \mathtt{send}(lecturer, login). \\
\quad CaWB :: recv(req, reply).\mathtt{if} \\
\quad (reply = allowed)?\mathtt{send}().\mathtt{out}.\mathbf{0} \\
\quad (reply = denied)?\mathtt{out}.\mathbf{0} \\
\mathtt{fi} \\
\mid \mathtt{recv}().\mathbf{0}
\end{array}\right) \tag{7}$$

```
1 ---> {ambient "St1" moves into ambient "cRoom"}
2 ---> {Sibling to sibling: St1 ===(St1,login)===> CaWB}
3 ---> {Parent to child: CaWB ===(St1,login)===> BB}
4 ---> {Child to parent: BB ===(St1,login,allowed)===> CaWB}
5 ---> {Parent to child: CaWB ===(St1)===> aList}
6 ---> {Sibling to sibling: CaWB ===(login,allowed)===> St1}
7 ---> {Local: St1 ===(logout)===> St1}
8 ---> {ambient "St1" moves out of ambient "cRoom"}
```

Fig. 4. Scenario 1 execution output

E. Other ambients

The *Display* ambient models the screen of the whiteboard. It receives documents such as lecture slides and tutorials and allows to visualising their contents on the screen. Its behaviour is specified in Eq. (8).

$$P_3 \;= !\mathit{CaWB}\uparrow\mathtt{recv}(\mathit{req},\mathit{reply}).\mathbf{0} \qquad (8)$$

The AVS ambient can be modelled in a similar way. It is assumed that this ambient is able to play audio and video contents received from the CaWB ambient; its behaviour is formalised in Eq. (9).

$$P_4 \;= !\mathit{CaWB}\uparrow\mathtt{recv}(\mathit{req},\mathit{reply}).\mathbf{0} \qquad (9)$$

The aList ambient maintains the list of attendance. It receives a user ID from the CaWB ambient and creates a new child ambient of name that user ID. This behaviour is modelled in Eq. (10).

$$P_5 \;= !\mathit{CaWB}\uparrow\mathtt{recv}(n).n[\mathbf{0}] \qquad (10)$$

Therefore the list of students attending a lecture is simply modelled inside the ambient aList as a process of the form $m_1[\mathbf{0}] \mid \ldots \mid m_\ell[\mathbf{0}]$, where m_i, $1 \leq i \leq \ell$ are distinct student ID numbers.

V. Validation

The CCA model of the context-aware whiteboard system (see Sect. IV) can now be executed using the CCA simulator ccaPL. A typical simulation output is depicted in Fig. 4, where the symbol "-->" represents an execution step; and the explanation of each execution step is given between a pair of curly brackets. For example the execution step in line 1 lets the ambient *St1* move in to the ambient *cRoom*; while the execution step in line 4 says that a child ambient *BB* sends the message (St1,login,allowed) to its parent ambient *CaWB*. The remaining execution steps can be explained in a similar manner. A series of scenarios will be used to validate the proposed model.

Scenario 1: this scenario is designed to check if any enrolled student is automatically logged in and added to the attendance list when the student enters the classroom. Equation (5) shows that only the lecturer *Lect1* and the student *St1* are initially enrolled on the module. Figure 4 depicts the execution output when the student *St1* enters the classroom.

The simulator ccaPL also displays the execution output graphically in the form of a diagram, which enables the user to visualise the concurrent behaviours of the system. Two types of diagrams can be produced: (i) a communication diagram depicting message passing between ambients; and (ii) a behaviour diagram showing both the movements of ambients and the communications between them.

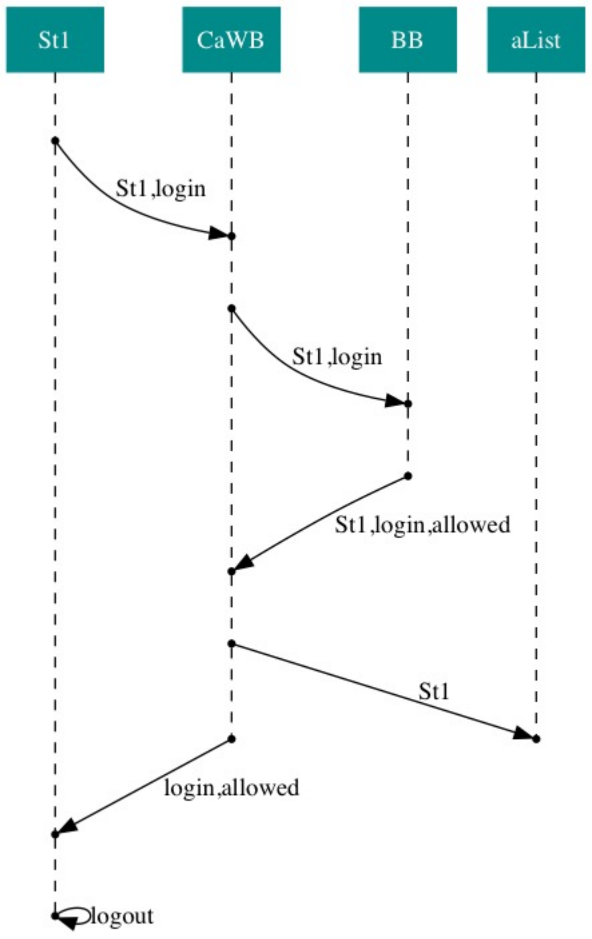


Fig. 5. Scenario 1 communication graph

The diagrams corresponding to the execution output in Fig. 4 are depicted in Fig. 5 for the communication diagram and Fig. 6 for the behaviour diagram. The top row in both diagrams lists the names of the ambients involved in the system being executed. The vertical dashed line represents the timeline for each ambient (time increases from top to bottom); and a solid directed line from the timeline of an ambient A say to that of an ambient B indicates a message passing from the ambient A to the ambient B at a specific time point, with the content of the message carried as a label to that line. Moreover, a behaviour diagram provides additional information about the mobility of ambients. A box labelled as "X --> Y" on the timeline of an ambient A indicates that the ambient A has moved from location X to location Y. Unlike the textual execution output like in Fig. 4, a behaviour diagram shows clearly the parallelism among of the involved ambients.

Scenario 2: This scenario shows that a lecturer teaching on a module can access the teaching materials on blackboard. The lecturer *Lect1* who is enrolled to teach on the module (see Eq. (5)) enters the classroom and requests access to teaching materials as per Eq. (6). The execution output is depicted in Fig. 7.

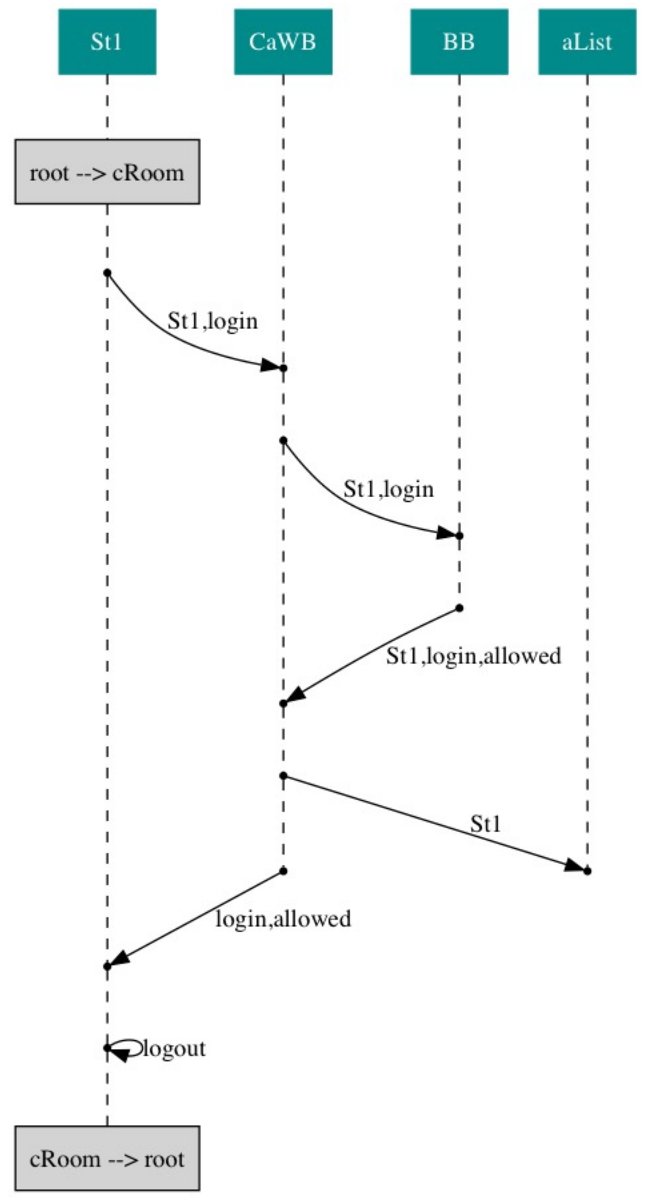


Fig. 6. Scenario 1 behaviour graph

```
1--->{ambient "Lect1" moves into ambient "cRoom"}
2--->{Sibling to sibling: Lect1 ==(Lect1,login)==> CaWB}
3--->{Parent to child: CaWB ===(Lect1,login)===> BB}
4--->{Child to parent: BB ==(Lect1,login,allowed)==> CaWB}
5--->{Parent to child: CaWB ==(Lect1)==> aList}
6--->{Sibling to sibling: CaWB ==(login,allowed)==> Lect1}
7--->{Sibling to sibling: Lect1 ==(Lect1,slides)==> CaWB}
8--->{Parent to child: CaWB ==(Lect1,slides)==> BB}
9--->{Child to parent: BB ==(Lect1,slides,slides_ppt)==> CaWB}
10--->{Parent to child: CaWB ==(slides,slides_ppt)==> Display}
11--->{Sibling to sibling: Lect1 ==(Lect1,audio)==> CaWB}
12--->{ambient "Lect1" moves out of ambient "classroom"}
13--->{Parent to child: CaWB ==(Lect1,audio)==> BB}
14--->{Child to parent: BB ==(Lect1,audio,mm01_wav)==> CaWB}
15--->{Parent to child: CaWB ==(audio,mm01_wav)==> AVS}
```

Fig. 7. Scenario 2 execution output

```
1--->{ambient "Lect2" moves into ambient "cRoom"}
2--->{ambient "St2" moves into ambient "cRoom"}
3--->{Sibling to sibling: Lect2 ===(Lect2,login)===> CaWB}
4--->{Parent to child: CaWB ===(Lect2,login)===> BB}
5--->{Child to parent: BB ===(Lect2,login,denied)===> CaWB}
6--->{Parent to child: CaWB ===(Lect2)===> aList}
7--->{Sibling to sibling: St2 ===(St2,login)===> CaWB}
8--->{Sibling to sibling: CaWB ===(login,denied)===> Lect2}
9--->{Parent to child: CaWB ===(St2,login)===> BB}
10--->{Child to parent: BB ===(St2,login,denied)===> CaWB}
11--->{ambient "Lect2" moves out of ambient "classroom"}
12--->{Parent to child: CaWB ===(St2)===> aList}
13--->{Sibling to sibling: CaWB ===(login,denied)===> St2}
14--->{ambient "St2" moves out of ambient "cRoom"}
```

Fig. 8. Scenario 3 execution output

Scenario 3: This scenario is designed to show that the model cannot allow students or lecturers who are not enrolled on the module to access the teaching resources of the module on the blackboard system. The lecturer *Lect2* and the student *St2* are not enrolled on the module. Figure 8 shows the execution output when they enter the classroom.

VI. Conclusion and Future Work

In this paper we have presented the formalisation of a context-aware whiteboard system in CCA. The system's specification was simulated using the execution environment of CCA called ccaPL on various scenarios. Some important properties of a classroom whiteboard system were validated as a proof of concept. future works, we will add more advanced functions to the system including listing absent students, recording lectures and uploading files on Blackboard.

References

[1] Al-Sammarraie, M, Siewe, F and Zedan, H (2011) Formal Specification of an Intelligent Message Notification Service in Infostation-based mLearning System using CCA. In *Proceedings of CCIT'11*, Dubai, UAE.

[2] Baldauf, M, Dustdar, S and Rosenberg, F (2007) A survey on context-aware systems. *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. 2, No. 4, pp. 263–277.

[3] Bardram, J (2005) The Java Context Awareness Framework (JCAF) – A service Infrastructure and Programming framework. In *Proceedings of the International Conference on Pervasive Computing*, Munchen, pp. 98–115.

[4] Atbaiga, N (2013) *The Next Generation of a Classroom White Board*. Unpublished dissertation (MSC), De Montfort University.

[5] Siewe, F (2011) *ccaPL: a Programming Language for the Calculus of Context-aware Ambients*. Technical Report, Software Technology Research Laboratory, De Montfort University, pp. 1–11.

[6] Siewe, F, Zedan, H, Cau, A (2010) The Calculus of Context-aware Ambients. *Journal of Computer and System Sciences*, 77(4), pp. 597–620.

[7] Weiser, M (1991) The Computer for the Twenty-First Century, *Scientific American*, 265(3), pp. 94–104.