Anonymous Agents Coordination in Smart Spaces

S. Balandin, I. Oliver, S. Boldyrev
Ubiquities Architectures team, Nokia Research Center
Itämerenkatu 11-13, 00180, Helsinki, Finland
{Sergey.Balandin, Ian.Oliver,
Sergey.Boldyrev}@nokia.com

A. Smirnov, A. Kashevnik, N. Shilov
Computer-Aided Integrated Systems laboratory, SPIIRAS
14-th Liniya 39, 199178, St.-Petersburg, Russia
{smir, alexey, nick}@iias.spb.su

***Abstract* – Rapid developments of communication, data processing and storage technologies, and continuing proliferation of consumer devices that surround user have created an opportunity for creation of a new generation of services based on smart spaces concept. The current approach for expanding mobile devices functionality is integration of new physical components. But this approach is bounded by the physical device size limits, dissipation of heat and the limited scalability of user experience due to small displays and incapability to produce high-end experience (e.g., audio) to the user. The smart spaces maximize the user benefits by utilizing capabilities of all available devices. This leads to a shift in the concept when instead of putting new functionality into the devices, all consumer electronics become a building blocks of the common information and service spaces. The smart spaces also provide another level of handling the user data. However, development of the smart spaces where a number of devices can use a shared view of resources and services is related to a number of problems. One of such problems is how to resolve possible conflicts arising from attempts of simultaneous access to the shared information. This paper describes an approach for coordination of anonymous agents, which solve this problem for the Smart-M3 smart space.**

***Keywords:* Smart Spaces; Use cases for consumer electronics; Smart-M3; Agents coordination; Shared information; Anonymous agents.**

I. INTRODUCTION

Modern device usage is moving towards so called “smart spaces” where a number of devices can use a shared view of resources and services [1], [2]. Smart spaces can provide better user experience by allowing the user easily integrate new devices into personal information infrastructures and allow seamlessly access all information distributed over the multi-device system from any of the devices. Examples of smart spaces can be found in [3], [4], [5]. One of the essential features assumed by such environment is the information sub-system that provides permanent robust infrastructure for storing and retrieving the information of different types from the multitude of environment participants.

Based on the analysis of earlier studies one can conclude that development of the Smart Spaces methodologies and techniques is a key requirement for creating attractive use case studies and building efficient developer eco-systems in the future. However, development of robust and efficient Smart Spaces solution is related to a need of addressing a number of practical problems. One of the problems to solve is coordination between the smart space participants, e.g., for resolving conflicts of simultaneous access to the shared data resource. To some extend this problem looks similar to the well known problem addressed in the database management systems, but after deeper study a lot of key differences could be identified. In computer science, the Atomicity, Consistency, Isolation, Durability (ACID) [6] is a set of properties that guarantee that database transactions are processed reliably.

The database modification procedure must follow the atomicity states, which implements “all or nothing” principle and refers to an ability to guarantee that either all of the transaction tasks are performed or none of them. Each transaction is said to be “atomic”, when if one part of the transaction fails, the entire transaction fails and the original state is preserved.

The consistency property ensures that the database remains in a consistent state before the start of the transaction and after its end (whether successful or not). It guaranties that only valid data could be written to the database. If for some reason, a transaction that violates the database consistency is executed, the entire transaction will be rolled back and the database will be restored to the last consistent state. On the other hand, every successfully executed transaction takes the database from one consistent state to another state that is also consistent.

The isolation refers to the requirement that other operations cannot access or see the data in an intermediate state during the transaction. This constraint is required to ensure good performance and guaranty inter-transactions consistency.

The durability is a guarantee that once the user has been notified about the success of the transaction, this state will persist. This means that the database must survive system failures and that the system already has checked the integrity constraints and won't need to abort the transaction. Many databases implement durability by writing all transactions into a transaction log that can be played back to recreate the system state right before a failure. In this case the new transaction can only be deemed committed after it is safely loaded into the log.

The well known and widely used in programming solution for restricting access to the shared resources is use of semaphores. The semaphore operations must be atomic, which means that no process may ever be preempted in the middle of one of those operations to run another operation on the same semaphore. There is a number of different implementation of semaphore principles, starting from a simple protected variable that locks/unlocks a certain resource and up to

 ISBN: 978-1-61208-100-7

counting semaphores which are the counters for a set of available resources, rather than a locked/unlocked flag of a single resource [7]. The semaphore value is a number of units of the resource that are free. If there is only one resource, a "binary semaphore" with values 0 or 1 is used.

Another solution used in concurrent programming is a monitor. The monitor is an object intended to be used safely by more than one thread [8]. The defining characteristic of a monitor is that its methods are executed with mutual exclusion. So for each point of time, at most one thread may be executing any of its methods. This mutual exclusion greatly simplifies reasoning about the implementation of monitors compared to the code that may be executed in parallel. The monitors also provide a mechanism for threads to temporarily give up exclusive access in order to wait for some condition to be met, and after that regain exclusive access and resuming their task. Monitors also have a mechanism for signaling to other threads that such conditions have been met.

However studying of the available solution has discovered that all of them are not suitable for the anonymous agent coordination in smart spaces, so the new solution has to be defined, which had been defined as a main target for this study. The next section provides an overview of the use case scenario that has been used as a main reference for studying the proposed solution. In Section 3 we present basic reference model of the discussed smart space. The method of resolving possible conflicts arising from simultaneous access to the shared information is described in Section 4. The following Section 5 gives a description of the developed demo prototype of the proposed solution for the reference use case scenario. The main results and findings of our study are summarized in Conclusion section.

II. SMART SPACE USAGE SCENARIO

The reference use case scenario describes a meeting taking place in a “smart room”, equipped with an intelligent whiteboard and a projector. The meeting participants have mobile devices (smartphones, PDAs, laptops, etc.) that store the appointments of the participants and their personal data, e.g., contact information, areas of interests, etc. Those meeting participants that are planning to make presentations have their presentations available on the mobile devices or accessible via internet/intranet (most important that the mobile devices always “know” how to access them).

In extension of the use case scenario defined in the previous works [9], this scenario is targeted in demonstrating the coordination function for resolving problems that arise due to possible simultaneous access to the shared information.

When the meeting participants entering to the room, their mobile devices discover the available smart space facilities, e.g., the whiteboard, and engage the handshaking protocol. If a participant wants to make a presentation, his/her mobile device is sharing the following information about the user: name, photo, domain of interests, e-mail, and phone number; and the presentation information: title, keywords, URI.

It is also necessary to schedule the presentations and create the meeting agenda. In this scenario the scheduling is done in the following simple way. There are several time slots covering whole time of the meeting. When a user comes, his/her presentation is scheduled into a free time slot. This is done by updating appropriate information units in the meeting room smart space, like it is illustrated by Figure 1.

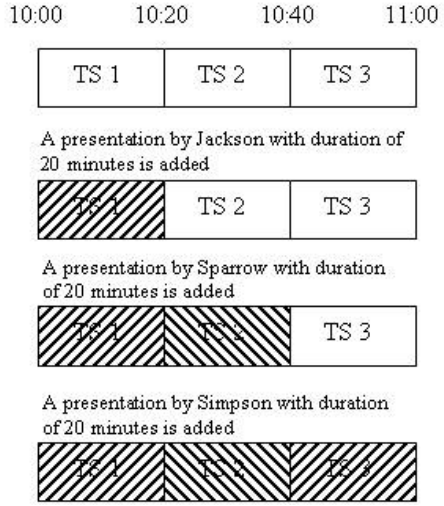

Figure 1. Scheme of scheduling presentations to the available time slots.

But the schedule conflicts can occur if two or more users simultaneously trying to schedule presentations (within resource request transmission and processing time delay) to the same time slot as it is shown in Figure 2.

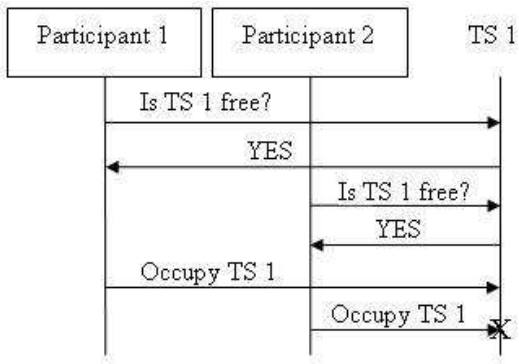

Figure 2. Possible conflict due to simultaneous access to shared information.

As a result, before the meeting starts the agenda is shown on the whiteboard including the speakers’ names and presentation titles. However, the same time slots can be occupied by different presentations or even some presentations will be lost from the list. The meeting participants can see the detailed agenda on the screens of their mobile devices, but agenda

 ISBN: 978-1-61208-100-7

might look differently for different people. The case can be even further complicated when some additional services are implemented, e.g., the presentation keywords could be translated to the preferred language (using the translator KP, which is also a part of the smart space), when the preferred language is taken from the user profile (the translator KP implements an interface to one of the Internet translation services). And the result M3 implementation will look like it is shown in Figure 3, where KP1 is a whiteboard, KP2 is a projector (PKP), KP3 is a translator and KP4…N are KPs of users' mobile devices (UKPs).

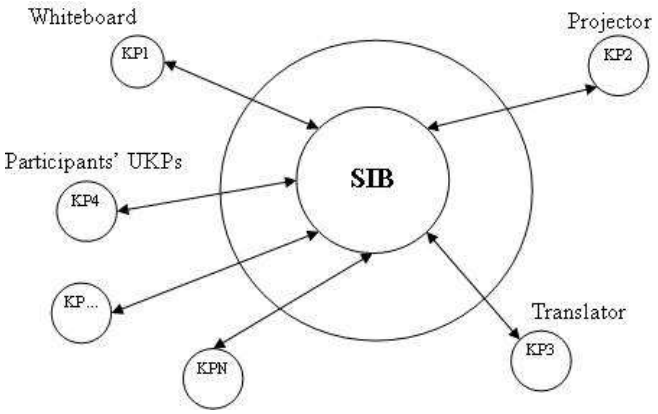

Figure 3. Current view of the proposed use case scenario.

Later in the paper we will show how this reference use case scenario can be implemented using the proposed coordination solution.

III. SMART SPACE REFERENCE MODEL

The general reference model of the discussed smart space could be illustrated by Figure 4.

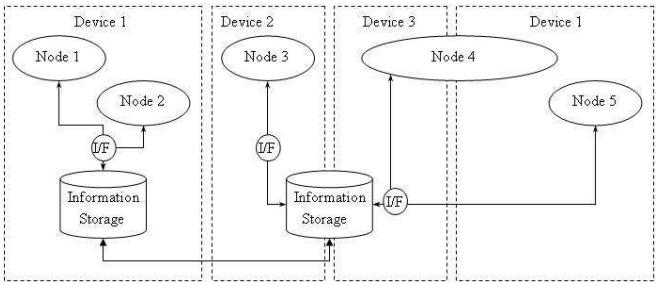

Figure 4. The reference model of discussed smart space.

Where:

Nodes - are logical elements capable to perform certain actions. One node can be distributed over several physical devices and several nodes can be located at the same device.

Information storages - also are logical units that store users information and can be distributed over several devices and several information storages can be located on at the same device.

I/F is an *interface* - that provides information exchange between the nodes and information storages. The interface is considered to be fully reliable and does not create additional delay and energy overheads. In this reference model the interface performs a technical function of connecting nodes to information storages. It does not implement logical functions and does not affect information transfer costs. For this reason the interface is not considered in the mathematical model.

Information is described by *information units* (IU) - represented as logical expressions: "subject"-"predicate"-"object" = [true | false], where *subject* is an actor (human or node that performs certain actions), *predicate* is an action that is being performed or supposed to be performed (e.g., "playing music") and *object* is what the action is performed with (e.g., a song being played). The nodes have predefined *behavior rules* defining their actions in line with the received information units.

From the implementation point of view the smart space can be illustrated as is shown in Figure 5.

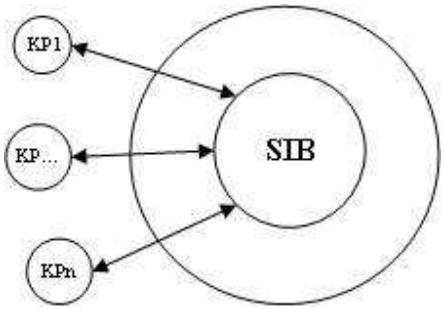

Figure 5. The Smart Space from implementation point of view.

The smart space itself consists of one or several Semantic Information Brokers (SIBs). The rules of information usage (applications) are implemented in knowledge processors (KP) connected to the smart space via SIBs. The SIBs are responsible for storing smart space information and its sharing: as soon as an information unit becomes available for the SIB, it becomes available for every KP. The knowledge processors are responsible for information processing.

IV. COORDINATION FOR CONFLICT RESOLUTION

So let's assume that we have a space that is used by 2 users. The users interact with the space by using their standard Knowledge Processors (KP), e.g., "u1" and "u2" correspondingly. If the "u1" needs to occupy certain resource R1, it currently only has to check that the statement: {"R1", "is_occupied_by", None} is valid, and if it is true then the KP "u1" can submit the triple: {"R1", "is_occupied_by", "u1"} to occupy it.

However, this works fine as long as we can guaranty that u1 and u2 will not try to simultaneously get access to the same resource, where term simultaneously is defined by the time interval from the moment when u1 has executed the first triple

 ISBN: 978-1-61208-100-7

and till the moment when it executes the second triple. But if during this time interval the node u2 will try to do the same, it also will get information that R1 can be occupied, which will result in resource access collision, as both nodes will have logical permission to occupy resource R1. As a consequence handling of the second triple becomes very complex procedure and independently of what tricks and fixes we will introduce at this stage with high probability it will lead to the logical errors and inconsistencies.

So in order to overcome the described above problem we introduce a special type of KP – the Coordinator KP. The Coordination KP acts as a kind of resource access manager. However, unlike classical resource manager solutions, which assume presence of a centralized application, to which all other application should send their resource requests, the functionality of Coordination KP is done based on principles described in the previous chapter. Most important that other applications do not need to know about presence of the Coordination KP in the space.

The coordination is performed seamlessly, automatically and anonymously by introducing a special set of RDF triples for handling access to the critical resources. Also the Coordination KP is subscribed to special triples that monitor all "resource access requests" and handles these requests on behalf of SIB, so that other KPs will not notice it. Below is the explanation how it works:

The Coordinator KP is subscribed to the information unit (RDF triple): {None, "check-insert", None}, where "None" logically means "any".

As a result, with the Coordinator KP the above scenario is changed as follows: the KP "u1" inserts the following rule into the smart space: {"R1, is_occupied_by, u1", "check-insert", "None"}, and subscribes to {"R1, is_occupied_by, u1", "check-insert-result", None}. The Coordinator KP checks the existence of the triple: {"R1", "is_occupied_by", None}. If it exists, the Coordinator KP inserts the triple {"R1, is_occupied_by, u1", "check-insert-result", "failure"}, the KP "u1" receives the result "failure" since it is subscribed. If the triple does not exist the Coordinator KP inserts the triple {"R1", "is_occupied_by", "u1"} and the triple: {"R1, is_occupied_by, u1", "check-insert-result", "success"}. The KP "u1" receives the result "success" since it is subscribed.

In case of simultaneous insert of rules by two KPs (u1 and u2): ("R1, is_occupied_by, u1", "check-insert", "None") and ("R1, is_occupied_by, u2", "check-insert", "None"), the Coordinator KP inserts the rule for the first KP and doesn't insert it for the second one. After that the KP u1 will occupy the resource R1 and the KP u2 will have to try to occupy some other resource, which can be offered to it by the Coordinator KP or defined internally by the KP u2. The result M3 smart space architecture with the Coordinator KP is presented in Figure 6, where the dotted lines show the information flow coming via the Coordinator KP.

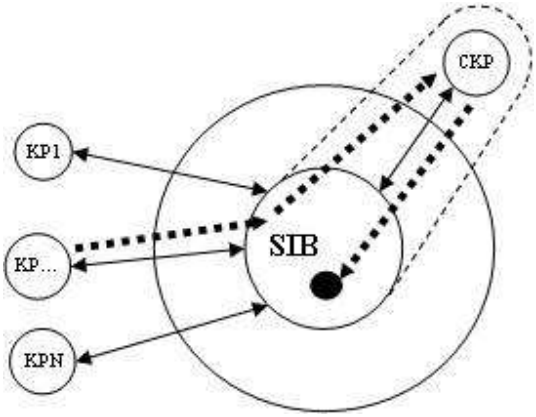

Figure 6. Organization of the information flow via Coordinator KP.

Further in-deep description of how the proposed solution can be implemented and used for the reference use scenario and the role of the Coordinator knowledge processor are discussed in the next chapter. Please also note that the same principle of coordination can be implemented completely inside the SIB.

V. IMPLEMENTATION OF THE SCENARIO

This scenario has been implemented using 6 personal computers (PC controlling the whiteboard, PC controlling the projector, PC controlling the Coordinator, PC controlling the Translator) and one Nokia N810 Internet Tablet emulating the user's mobile device. The other two user mobile devices were emulated on PCs. A proprietary M3/Piglet toolkit has been used for the prototype development. The knowledge processors were implemented using Python programming language.

Figures 7–10 are the screenshots for different knowledge processors at different stages of the scenario, e.g., Figure 7 shows work of the KP installed on the MAEMO device of the first meeting participant, which presentation was assigned to the first time slot. Figure 8 reflects work of the KP of the third meeting participant. The presentation was assigned to the third time slot. One can also see the translations of the presentation keywords, which were translated into Finnish language. The execution log of the Coordinator KP is shown in Figure 9. It lets the UKP of the first participant to occupy the time slot TS 1, the second participant to occupy the time slot TS 2, and the third participant to occupy the time slot TS 3.

The result output for the whiteboard KP is shown in Figure 10. One can see how the agenda is built based on users entering the room and occupying presentation time slots. Then, during the meeting the current presentation is highlighted (with three asterisks).

 ISBN: 978-1-61208-100-7

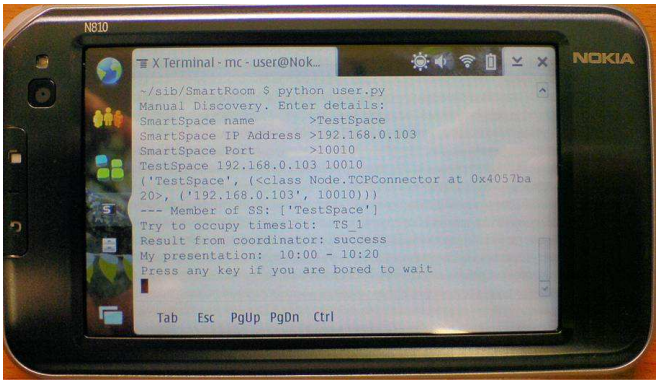

Figure 7. Screenshot of UKP running on Nokia N810 MAEMO device.

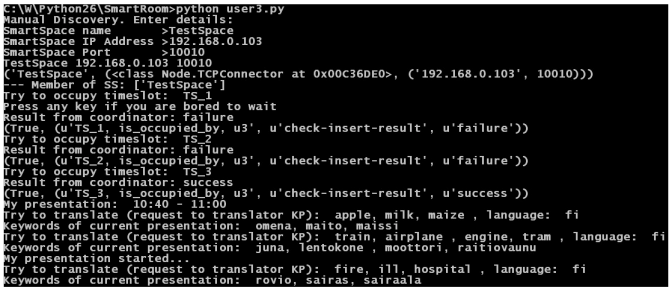

Figure 8. The status window of KP on PC of the third meeting participant.

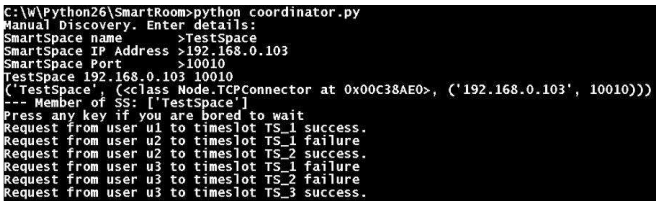

Figure 9. Log-output window of the Coordinator knowledge processor.

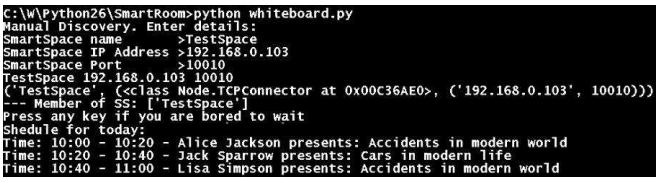

Figure 10. Log-output window of the Whiteboard knowledge processor.

As a result, nowadays we have full implementation for all of the above described elements that allow performing anonymous agents' coordination in Smart Spaces.

VI. CONCLUSIONS

The paper describes a solution for anonymous coordination of the agents, which allows addressing and solving a huge set of problems arising from a possibility of simultaneous access to the shared information.

The existing mechanisms for solving similar problems, such as transactions (used in database management systems), semaphores and monitors (used in programming) could not be applied directly. As a result an additional coordinator knowledge processor implementing the required functionality was introduced and described in the paper.

The paper gives detailed description of this knowledge processor work principles, which are also illustrated using an example implementation of the proposed principle for the reference case study scenario.

ACKNOWLEDGMENT

This paper is done within scope of the joint project between St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS) and Nokia Research Center. Some of the results are due to research carried out as a part of the project funded by grants # 09-07-00436-a and 08-07-00264-a of the Russian Foundation for Basic Research, and project # 213 of the research program "Intelligent information technologies, mathematical modelling, system analysis and automation" of the Russian Academy of Sciences. The authors would like to thanks Nokia, Russian Foundation for Basic Research and Russian Academy of Sciences for the provided financial support.

REFERENCES

[1] Oliver, I. and Honkola, J. *Personal Semantic Web Through A Space Based Computing Environment*, Middleware for Semantic Web 08, proceedings of ICSC'08, 2008.

[2] Oliver, I., Honkola, J., and Ziegler, J. *Dynamic, Localised Space Based Semantic Webs*, WWW/Internet Conference, Freiburg, Germany, 2008.

[3] Oliver, I. *Design and Validation of a Distributed Computation Environment for Mobile Devices*, proceedings of European Simulation Multiconference: Modelling and Simulation, 2007.

[4] Oliver, I., Nuutila, E., and Seppo, T. *Context gathering in meetings: Business processes meet the Agents and the Semantic Web*, proceedings of the 4th International Workshop on Technologies for Context-Aware Business Process Management, 2009.

[5] Jantunen, J., Oliver, I., Boldyrev, S., and Honkola, J. *Agent/Space-Based Computing and RF memory Tag Interaction*, proceedings of the 3rd International Workshop on RFID Technology - Concepts, Applications, Challenges, 2009.

[6] Wikipedia: http://wikipedia.org/wiki/ACID. Retrieved: 16.7.2010.

[7] Dijkstra, E., *Cooperating Sequential Processes*, Technical Report EWD-123, Technological University, Eindhoven, Netherlands, 1965.

[8] Hoare, C. *Monitors: an operating system structuring concept*, Communications of the ACM, Vol. 17 No. 10, pp. 549–557, 1974.

[9] Oliver, I., Nuutila, E., and Törma, S. *Context gathering in meetings: Business processes meet the Agents and the Semantic Web*, proceedings of the 4th International Workshop on Technologies for Context-Aware Business Process Management (TCoB 2009), 2009.

 ISBN: 978-1-61208-100-7