Flexible Analysis of User Actions in Heterogeneous Distributed Learning Environments

Lars Bollen, Adam Giemza, and H. Ulrich Hoppe

Department of Computer-Science and Applied Cognitive Science
University of Duisburg-Essen, Germany
{bollen,giemza,hoppe}@collide.info

Abstract. In this paper, we describe an architectural framework for the engineering of distributed learning environments with different devices and multi-language agent support. The framework consists of a central Tuple Space server and clients that differ in hardware (PDAs, PCs with projection) and in programming languages (C#, Prolog, Java). The analysis components use state patterns and action patterns to be defined in and interpreted by Prolog. This framework has been used for supporting the design rationale method QOC in a collaborative visual modelling environment.

Keywords: Collaboration analysis, mobile devices, tuple spaces, design rationale, discussion support.

1 Introduction

This paper will explore technical issues relevant for the orchestration of classroom settings with interactive devices. With new mobile and wireless technologies, there is growing variety of options in terms of different types of devices that can be used for this. Both the cost and form factors support the feasibility of a one-to-one relation between learners and orchestration and interactive devices (cf. [1]). However, it is questionable if classroom orchestration can (or should) essentially rely on one type of computational device. Liu and Kao [2] have studied classroom interaction patterns with different combinations of devices, including tablet PCs and big interactive screens. These studies corroborate the hypothesis that the use of personalised devices combined with a single public interactive display is clearly superior to using only personalised devices (though with shared content). One of the reasons for the superiority mixed setting is the *lack of shared visual focus* with small personal devices only. This lack of shared visual focus is seen as a cause of fragmented communication observed in a classroom study. The Liu and Kao study provides details about communication patterns and micro activities, such as eye contact and hand/finger pointing, leading to the conclusion that "shared displays enable group members to participate closely in shared activities and establish ideal communication patterns".

In accordance with these plausible findings of Liu and Kao, we have designed dedicated classroom scenarios with heterogeneous device orchestration and "functional differentiation" exploiting the advantages of different device types. One of the results was the "Mobile Notes" system to support classroom discussions in which PDAs are used essentially as input devices in combination with a large interactive screen providing a graph structured visual representation [3]. The original version of "Mobile Notes" used a simple architecture with a relational database for buffering the contributions (text or simple sketches) prepared on the PDAs "waiting" to be inserted into the discussion graph. The discussion moderator could view the buffered messages and decide if and possibly when they should be inserted. In our recent work, we have further developed this scenario by adding the following features: (1) support for a specific kind of structured representation used in collaborative design scenarios, and (2) improved awareness for the moderator based on machine analysis of contributions. The solution is based on a more sophisticated distributed and heterogeneous architecture using different implementation languages and agent plug-ins for the analysis. The principles of analysis are based on a previously developed framework described in Gassner et al. [4].

In the sequel, we will describe the implementation platform, the specific scenario and the underlying architecture including aspects of knowledge engineering for intelligent support and a first study to evaluate the usefulness and ease-of-use of this environment.

2 The Use of Tuple Spaces as a Distributed Computing Platform

The Tuple Space approach is based on a blackboard architecture and was introduced together with the coordination language "Linda" [5]. It can serve different coordination and communication functions using a central server which acts as the blackboard and holds all messages or "entries". The clients solely exchange messages with the server, there is no direct client-to-client connection. Entries are in "tuple" format, i.e. they consist of ordered lists of fields containing primitive data. So the server can be seen as a kind of tuple exchange place used as a shared working memory by the clients. A more recent implementation of Tuple Spaces is part of the Java Jini framework originally developed by Sun Microsystems in 1998 under the name of "JavaSpaces". In addition to the standard read/write operations, JavaSpaces has also introduced a *notification mechanism*. Another extension of the original idea is the concept of *leases* to assign a limited life time to entries. Almost simultaneously, another Java based Tuple Space implementation called TSpaces has been developed and published by IBM's Almaden Research Center [6].

The Tuple Space idea has recently gained attention for the purpose of designing and implementing cooperative environments: *Group Scribbles* [7] developed at SRI is a collaborative environment for sharing graphical and textual notes ("scribbles"). Group Scribbles uses TSpaces to support synchronous co-construction with shared and private workspaces and is available for different hardware platforms. A second relevant application of Tuple Spaces for groupware was developed in the *Amenities* project [8]. It uses JavaSpaces to synchronise appointment books of academic researchers. A main focus of this approach is on the provision of rich awareness functions using a Tuple Space architecture with synchronised views, user lists and remote references.

We will particularly draw on the work of Giemza et al. [9], in which Tuple Spaces are used as a general platform for engineering distributed cooperative systems with agent support. By providing standardised clients for different programming languages (such as Java, C#, Ruby or Prolog), the Tuple Space can also serve as a "language switchboard". This makes the provision of specific one-to-one interfaces between different language environments obsolete. For example, a C# client will connect any PDA environment to the shared memory, whereas an intelligent query agent may be formulated in Prolog and the connection to a learning platform will be provided through a PHP client.

The following features of this architecture and approach have turned to be of specific relevance and benefit:

- Once a TS-client interface is provided for a certain language, no more syntactic interfacing is needed for a specific application.
- The blackboard communication pattern provides a persistent data store and "data exchange service" for distributed applications and allows for a high degree of independence (loose coupling) in the development of the different system components.
- Language heterogeneity supports an effective way of specialization in a programming team.

There are multiple reasons for using the one or the other implementation language in nowadays application systems. An essential point to consider is certainly the adequacy of the language for the problem at hand. E.g., Prolog may be considered as particularly well suited for the analysis of complex objects with a symbolic representation, whereas Java may be seen as the language of choice for distributed applications with visual interfaces. But also the availability and support of a language on a specific class of devices may be an issue. In the case of PDAs, we have recently moved from using Java (J2ME) to C# for various (mainly practical) reasons. From a knowledge engineering point of view, this approach allows for defining collaboration patterns without having to deal with low level interfacing problems. In the rest of the paper these advantages will be explained in more detail based on the existing implementation.

3 The Application Scenario

3.1 Motivation

Recently, we have described an approach to support face-to-face discussions in classrooms or meeting rooms in which PDAs would not be used to share information but to provide individual input in the form of small notes or sketches [2]. In this *Mobile Notes* scenario, the information to be shared by the discussion group is provided on a big interactive screen or whiteboard. Now, we have specialised the quite general *Mobile Notes* scenario to a more specific structured representation of the discussion content. This representation (QOC) has its origin in the area of "design rationale" methods [10] (see also section 3.2). In our case it supports the externalization of a decision process.

The fact that we have a more specialized application and a more structured representation than in the *Mobile Notes* scenario allows us to analyse the discussion content and process based information on certain temporal and structural patterns. In this context, the blackboard communication model is used to plug in the corresponding analytic agents. Fig. 1 depicts a typical setup of this learning environment.

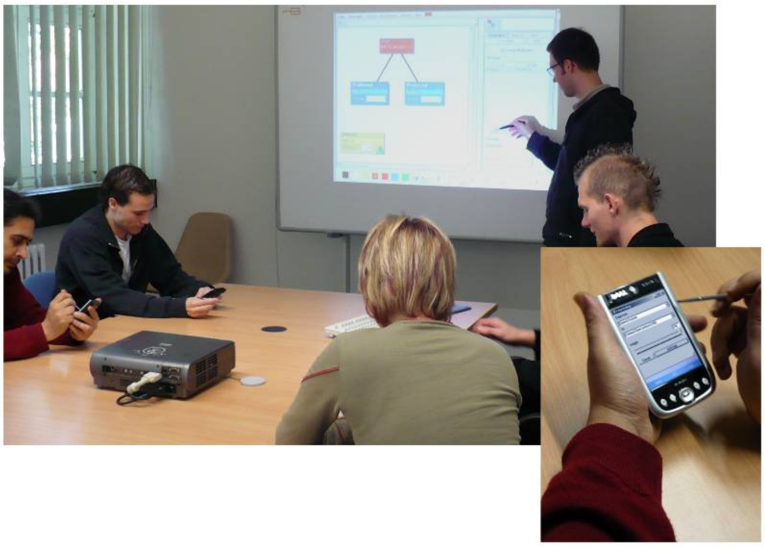


Fig. 1. The application scenario

3.2 QOC Based Design Decision

The concept of “design rationale” comprises a variety of methods to support structured decision making and externalization in the design of (technical) artefacts (cf. [10]). Design rationale techniques have been particularly applied and developed in the field of GUI and interaction design. Support technologies for design rationale approaches include the provision of machine readable representations (to be shared by humans) as well as interactive/cooperative media arrangements to facilitate communication and flow in the design process.

“Questions - Options - Criteria” or QOC [11] is in first place a structured representation to be used in a design rationale process, starting with a specific question or design issue to be resolved, than stating alternative options and introducing criteria for a comparative, weighted evaluation of the options. The final product is a QOC graph which would document and explain the specific decision (see Fig. 5).

The QOC representation is supported by a specific “palette” in our collaborative visual modelling environment *FreeStyler* [12]. It has been used quite successfully in classrooms as well in software development projects by students.

The specific application scenario to be supported here is the use of QOC in a face-to-face setting in which the shared visual representation is displayed on an interactive board under the control of a teacher or moderator. PDAs are used by the participants to make input to this shared model. The results of the analytic agents are fed back to the public display as a kind of content oriented awareness information.

3.3 Discussion Moderation

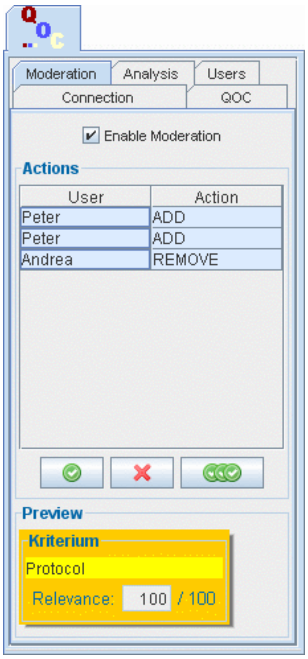
Additionally, the architecture described in the following chapter allows for the moderation of incoming students' contributions.

In the paper at hand, the term *moderation* describes the possibility of a moderator or teacher to preview and approve or reject students' actions. Single actions in our case are

- Adding a question, option or criterion
- Deleting or changing the content of one of the above
- Adding or removing an edge
- Changing the weight of an edge.

If one of these actions is conducted by a student, it will appear in the moderation interface in the *FreeStyler* environment that is under control of the moderator (see Fig. 2). Here, all actions are collected in a list and can be previewed. The moderator can choose to approve an action (i.e. the action is executed in the StateSpace (see next chapter)) or it can be rejected (i.e. the action is not executed and will be removed from the moderator's panel).

Of course, the moderation feature can be switched off completely. In that case, all students' actions are executed directly without further interference or delay.



☑ Enable Moderation

User	Action
Peter	ADD
Peter	ADD
Andrea	REMOVE

Kriterium
Protocol
Relevance: 100 / 100

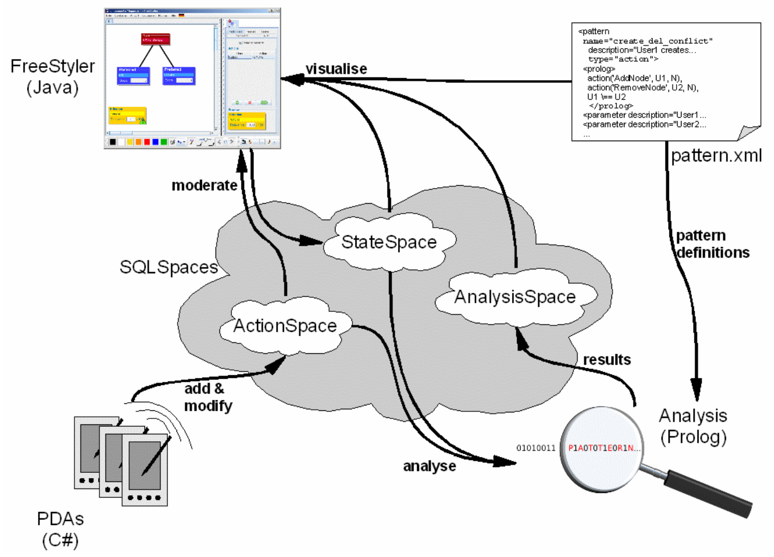
Fig. 2. Moderation interface with action preview

4 Overall Architecture

The overall architecture consists of the QOC-FreeStyler application shown on the public display, several clients that run on PDAs, an analysis engine (Prolog) and a Tuple Space (TS) server as communication and synchronisation platform (see Fig. 3). For the TS server, we use our own implementation of SQL Spaces [9].

The TS consist of several spaces that play a central role in this architecture. The *StateSpace* holds a representation of the current QOC model that is visualised by the FreeStyler application. The PDA clients may perform actions to modify or add to the model, which will be written into the *ActionSpace* first. At that point, as described in the previous chapter, actions can be moderated (i.e. previewed and possibly approved by a moderator) or conducted directly. The separation of actions and states does not only allow for an easy implementation of such moderation, but it provides a convenient way for action analysis, since all actions are persistently available to the Prolog engine (see next chapter).

The analysis engine continuously tries to match given patterns to the contents of the *StateSpace* and *ActionSpace*. If a pattern matches, the result is written to the *AnalysisSpace*, from which it can be retrieved and visualised by FreeStyler.



<pattern
 name="create_del_conflict"
 description="User1 creates...
 type="action">
<prolog>
 action('AddNode', U1, N),
 action('RemoveNode', U2, N),
 U1 \== U2
 </prolog>
<parameter description="User1...
<parameter description="User2...
...

Fig. 3. Overall architecture

5 Knowledge Engineering of Cooperation Patterns

Fig. 4 shows the knowledge engineering process of cooperation patterns. A knowledge engineer uses a Prolog testing environment to create and test Prolog predicates that match interesting patterns in user actions or model states (or even a mixture of both). Once the knowledge engineer finishes the pattern creation process, the predicates are annotated with descriptions about the meaning of the patterns and of the parameters used. This information is later used by the *Awareness Display* to convey information about found patterns to the end user at run time.

The following lines are an excerpt from a *pattern.xml* file that is used by the *Awareness Display*.

```
<pattern
  name="create_del_conflict"
  description="User1 creates a node and a different User2
               deletes it"
  type="action">
  <prolog>
    action('AddNode', U1, N),
    action('RemoveNode', U2, N),
    U1 \== U2
  </prolog>
  <parameter description="User1" var="U1"/>
  <parameter description="User2" var="U2"/>
  <parameter description="Node" var="N"/>
</pattern>
```

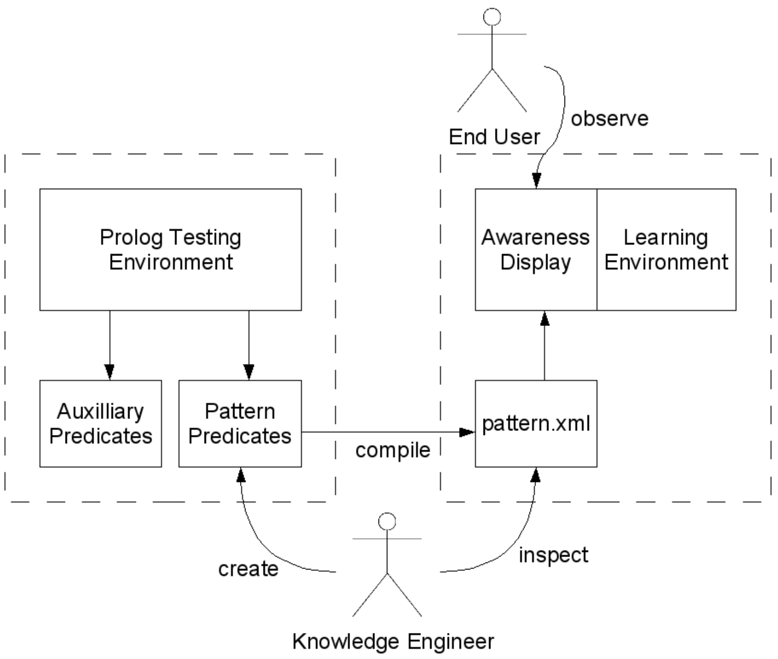


Fig. 4. Knowledge engineering of cooperation patterns

One advantage of this kind of knowledge engineering framework is the homogeneity of the interface. In the approach at hand, the knowledge engineer can create, test and describe analysis patterns in one single environment (i.e. a Prolog programming environment like SWI-Prolog[1]). He does not have to leave his familiar working environment. Additionally, by having the *pattern.xml* file, analysis patterns, their descriptions and metadata can be exchanged between users and moderators by simply copying one file.

6 Action Patterns and State Patterns

As stated above, the described framework allows for performing analyses based on information about the actual state of a model as well as analyses based on the history of user actions.

Typically, *state patterns* deal with syntactic and semantic features and characteristics of the used modelling language. These kinds of patterns describe certain constellations of nodes, edges and their attributes. For the QOC method, we identified some patterns that are potentially interesting for a moderator:

- The model contains an unconnected criterion;
- The model contains an option with no attached criteria;

[1] See http://www.swi-prolog.org, 14th April 2008.

- There is no preferred option in the model;
- A criterion has equal effects to all options;
- ... and others.

Action patterns deal with certain sequences in the history of user actions. Typically, these patterns describe some kind of significant and possibly interesting behaviour of single users or they describe occurrences of collaboration between two or more users. These patterns tend to be more domain independent than the state patterns, so that they can potentially be used with various modelling languages and learning applications.

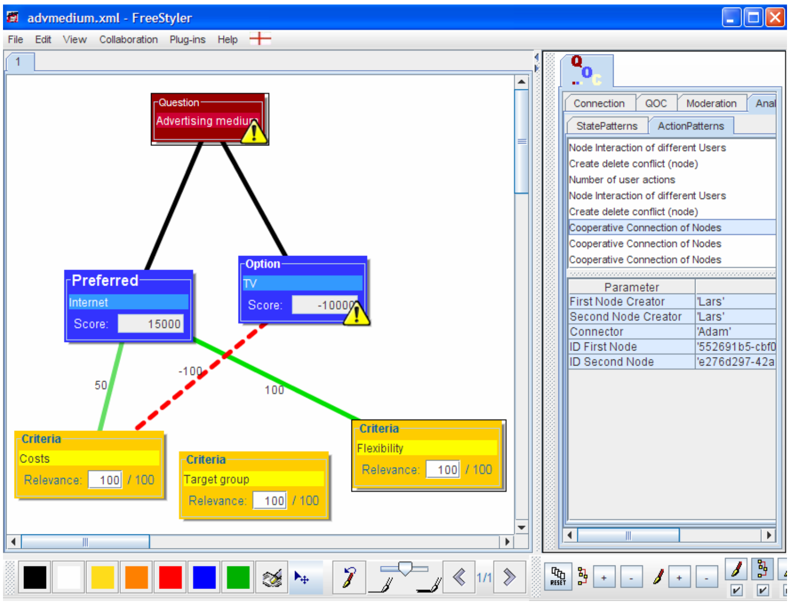
Examples for action patterns are:

- One user creates an object, a different user deletes this object (conflict?).
- One user creates an object, a different user connect this object (collaboration?).
- One user is clearly doing most of the actions (dominating others?)

However, there are also domain dependent action patterns. E.g., for QOC, a series of actions that change the weight of an edge from positive to negative values can be quite significant and could not be detected by state patterns only.

Additionally, combinations of state and action patterns are feasible and meaningful, too. Imagine having a pattern that detects a characteristic situation in the model (state analysis) in combination with searching for possibly collaborative user actions that lead to this situation (action analysis).

Fig. 5 shows a screenshot of the FreeStyler application and the awareness component notifying the moderator of a matched pattern.



Node Interaction of different Users
Create delete conflict (node)
Number of user actions
Node Interaction of different Users
Create delete conflict (node)
Cooperative Connection of Nodes
Cooperative Connection of Nodes
Cooperative Connection of Nodes

Parameter	
First Node Creator	'Lars'
Second Node Creator	'Lars'
Connector	'Adam'
ID First Node	'552691b5-cbf0
ID Second Node	'e276d297-42a

Fig. 5. QOC model and awareness display for a “Cooperative Connection of Nodes”

7 Evaluation of MobileQOC

The MobileQOC environment has been evaluated for their usefulness and ease-of-use in a study conducted in four single sessions with 19 participants altogether. In each session, one participant took the role of a moderator and four participants took the role of a discussant (one person did not show up, thus 19 persons).

In each session, the participants were given a design decision task that should be solved by using the MobileQOC environment. The setup of this study has been very similar to Fig. 1.

The four sessions have been evaluated by using questionnaires, interviews and observations. The questionnaires were built and evaluated by using the Technology Acceptance Model (TAM) by Fred Davis [13]. From the Technology Acceptance Model, the two most prominent aspects *Perceived Usefulness* and *Perceived Ease of Use* have been used in this study. The questionnaires have been designed using a 7 scale Likert scale and consisted of the following items:

Perceived Usefulness questionnaire items:

1. I think MobileQOC could improve my performance in design decisions in school or university courses.
2. MobileQOC could help me to accomplish design decisions more quickly.
3. I think using MobileQOC enhances my effectiveness in design decisions.
4. I think using MobileQOC decreases my productivity in school or university courses.
5. Using MobileQOC makes it easier to carry out design decisions.
6. I think MobileQOC slows down design decisions.
7. Overall, I find the MobileQOC application useful in school or university courses.

Perceived Ease-of-Use questionnaire items:

1. The MobileQOC application is easy to learn.
2. MobileQOC is rigid and inflexible to interact with.
3. MobileQOC is easy controllable and behaves as expected.
4. My interaction with the MobileQOC application is easy for me to understand.
5. It is hard for me to remember how to perform tasks using the MobileQOC application.
6. Interacting with MobileQOC is mentally exhausting.
7. Overall, I find the MobileQOC application easy to use.

Besides the TAM questionnaires, each session has been observed by a non-participating observer and the moderators have been interviewed on their use and estimation of the moderation and analysis features of the MobileQOC environment.

7.1 Questionnaire Results, Interview Conclusions and Observation Findings

The reliability of the questionnaire results can be considered high: The Cronbach's Alpha value for the Perceived Usefulness items (PU) is .927; the value for the Perceived Ease-of-Use items (PEU) is .776.

The results of the single Perceived Usefulness items show high values in all (non-reversed) items (the mean ranges from M=4.47, SD=1.727 to M=5.4, SD=1.298 on a 7 level Likert scale). Remarkably, the summarising item 7 gets the highest mean value.

The results for the Perceived Ease-of-Use items show even better results: The mean values of the non-reversed items range from M=6.0, SD=1.0 M=6.8, SD=.561 while the reversed items range from M=1.27, SD=.594 to M=3.2, SD=1.32.

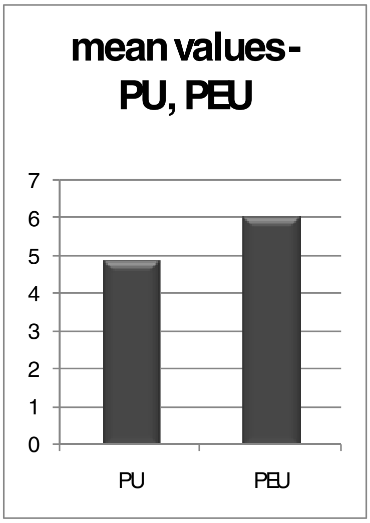


Fig. 6. Mean values of PU and PEU over all cases

Fig. 6 finally shows the mean values for the Perceived Usefulness and Perceived Ease-of-Use over all items and cases.

Overall, the usefulness and the ease-of-use can be considered as perceived well for the MobileQOC environment.

From the interviews, you could learn that the moderation features (i.e. accepting or rejecting incoming participants' action) is perceived as of little use in sessions with a small number of participants and with "well-behaving" groups, but estimated to be of high value when moderating large groups, several groups at a time or when groups are showing destructive or socially bad behaviour. The analysis features for state patterns (that mostly find flaws and shortfalls in the QOC model) are generally considered highly useful, while the action patterns (that generally detect specific collaborative features) are considered less useful for the same reasons as the moderation feature.

The observation showed that the participants' attention was not completely caught by the mobile devices or by the projected display, but they talked with each other to discuss particularities on the domain level ("What do you think the edge weight should be?") and on an organisational and social level ("You both connect these nodes [pointing to whiteboard] and we connect these nodes [pointing to whiteboard], so that everybody has to do something now..."). This goes well along with the findings of Liu and Kao [2], that appraise large shared displays as supportive for the externalization and articulation of student thinking.

8 Discussion and Outlook

In the previous chapters, we presented a learning environment to support design rationale discussion using the QOC method. Mobile devices serve as input devices to give students an opportunity to participate freely and unhindered from peers. The graph-based modelling application FreeStyler is used to collect, organise and display the contributions on a large whiteboard, acting as a shared visual focus. The paper concluded with presenting the results of a study to evaluate the usefulness and users' estimation of MobileQOC.

Moderation and analysis features support a moderator or teacher to be able to balance discussions and take care for syntactically sound models.

On a technical level, the environment bases on Tuple Spaces to provide a powerful and flexible architecture that easily integrates various programming languages, different devices and allows for sophisticated analysis components.

In further developments, we plan to generalise and extend this approach and architecture to various modelling languages like Petri Nets, System Dynamics or UML. Pre-defined Prolog predicates that are useful and meaningful for state analysis and actions analysis in various languages can be identified in this process of generalisation.

Additional agents will be used to further process the content of the *Analysis Space* in order to generate direct feedback in the form of recommendations or adaptations of the environment (instead of just displaying awareness information).

Furthermore, we plan to include the actions in the moderation queue (i.e. actions that have been committed by users but not yet approved by a moderator) into the analysis cycle. Thus, a moderator would be able to assess the impact of user actions before they are released.

References

1. Chan, T., Roschelle, J., Hsi, S., Kinshuk, M.S., Brown, T., Patton, C., Cherniavsky, J., Pea, R., Norris, C., Soloway, E., Balacheff, N., Scardamalia, M., Dillenbourg, P., Looi, C., Milrad, M., Hoppe, U.: One-to-one technology-enhanced learning: An opportunity for global research collaboration. Research and Practice in Technology Enhanced Learning 1(1), 3–29 (2006)
2. Liu, C., Kao, L.L.: Handheld devices with large shared display groupware: tools to facilitate group communication in one-to-one collaborative learning activities. In: Proceedings of IEEE WMTE 2005, Tokushima, Japan, March 2005, pp. 128–135 (2005)
3. Bollen, L., Juarez, G., Westermann, M., Hoppe, H.U.: PDAs as input devices in brainstorming and creative discussions. In: Proceedings of 4th International Workshop on Wireless, Mobile and Ubiquitous Technologies in Education (WMUTE 2006), pp. 137–141. IEEE Computer Society Press, Los Alamitos (2006)
4. Gassner, K., Jansen, M., Harrer, A., Herrmann, K., Hoppe, H.U.: Analysis methods for collaborative models and activities. In: Proceedings of CSCL 2003, Bergen, Norway, June 2003, pp. 369–377 (2003)
5. Gelernter, D.: Generative communication in Linda. ACM Transactions on Programming Languages and Systems 7(1), 80–112 (1985)
6. Lehman, T.J., McLaughry, S.W., Wycko, P.: T Spaces: The Next Wave. In: Proceedings of the 32nd Hawaiian International Conference on Computer Systems 1999, HICCS, Maui, Hawaii (1999)
7. Brecht, J., DiGiano, C., Patton, C., Tatar, D., Chaudhury, S.R., Roschelle, J., Davis, K.: Coordinating networked learning activities with a general-purpose interface. In: Proceedings of the 5th World Conference on Mobile Learning, Banff, Canada (October 2006)
8. Garrido, J.L., Noguera, M., Gonzalez, M., Gea, M., Hurtado, M.V.: Leveraging the Linda coordination model for a groupware architecture implementation. In: Dimitriadis, Y.A., Zigurs, I., Gómez-Sánchez, E. (eds.) CRIWG 2006. LNCS, vol. 4154, pp. 286–301. Springer, Heidelberg (2006)

9. Giemza, A., Weinbrenner, S., Engler, J., Hoppe, H.U.: Tuple spaces as flexible integration platform for distributed learning environments. In: Proceedings of ICCE 2007, Hiroshima, Japan, November 2007, pp. 313–320 (2007)
10. Regli, W.C., Hu, X., Atwood, M., Sun, W.: A survey of design rationale systems: Approaches representation, capture and retrieval. Engineering with Computers 16, 209–235 (2000)
11. MacLean, A., Young, R., Belloti, V., Moran, T.: Questions, options, and criteria: Elements of design space analysis. Human-Computer Interaction 6(3/4), 201–250 (1991)
12. Hoppe, H.U., Gaßner, K.: Integrating collaborative concept mapping tools with group memory and retrieval functions. In: Proceedings of CSCL 2002, Boulder, USA, January 2002, pp. 716–725 (2002)
13. Davis, F.D.: Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly 13(3), 319–340 (1989)