Model-Based Development of Synchronous Collaborative User Interfaces

George Vellis
Department of Applied Information Technology & Multimedia
Technological Education Institution of Crete, Stavromenos 71004, Heraklion, Crete, Hellas
epp646@venus.cs.teiher.gr

ABSTRACT

This paper undertakes with collaborative software development taking into account requirements emerged from recent progress in technologies relevant to networks and computing devices. Considering this technological breakthrough, especially under the light of the consequently sharply growing online virtual communities, we can deduce that a new substance is given to the software supporting collaborative practices for multiple environments. In such cases, one important aspect to consider is the user interfaces (UIs) design supporting group work appropriately. The results today offer a rich insight to the desired groupware functionality and the features devised to facilitate such functionality (i.e., replication models, object sharing, floor control, etc). On the other hand, very little is known about their capability to facilitate generation of multi-user interfaces to groupware applications. With the advent of model-based user interface engineering, which signifies a move towards transformation-based approaches to generating the user interface, one challenge is bridging across these two perspectives. The current work seeks to contribute to this goal by identifying the type of models needed to capture collaborative behavior in synchronous multiple user interface settings as well as generating the collaborative user interface by making use of suitable platform-oriented architectural models.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques – *Computer-aided software engineering (CASE), User interfaces*. H.5.2 [**Information Interfaces and Presentation**]: User Interfaces - *Graphical user interfaces (GUI).*

H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces - Collaborative computing, Computer-supported cooperative work.

General Terms: Design, Human Factors.

Keywords: Synchronous groupware, Model-Based UI development, User interface description Languages (UIDL), Multi-User Interfaces.

1. INTRODUCTION

The growth of network technologies and ubiquitous computing brought and incorporated information technology deeply into our daily practices [23]. The above motivate the manifestation of software systems, capable of operating in multiple platforms / environments (i.e.: PCs, PDAs, hand-held computers, tabletPC's, etc.) while facilitating transparent information exchange and collaboration among their potential users. In terms of end user experience, this trend continues to pose new challenges. One particular challenge relates to the design and development of user interfaces (UIs) fostering synchronous collaborative social interaction in ubiquitous contexts.

Retrospectively; since the early STAR user interface, and for more than three decades now, the prime user interface development technique has been toolkit programming. Although, alternative user interface development methods have emerged making use of abstract notations and mark-up languages to facilitate mapping of abstract components to platform-specific toolkit libraries by delegating the display to a platform-specific renderer [14]. Thus far, the efforts devoted to using model-based engineering to build collaborative interfaces are very few and limited in the scope of collaborative behavior supported. Indicative examples include AMENITIES[8], CIAM[28], TOUCHE[21], and FlowiXML[9]. These efforts concentrate primarily on devising notations and tools to model cooperative behavior and workflows. In effect, their primary contribution is that they make explicit different elements of collaboration (i.e., roles, responsibilities and tasks) using dedicated notations such CTT or the Co-interaction Diagram in [20]. Only some of the existing efforts make inroads towards generating the user interface of collaborative applications. An example of this work is FlowiXML. However, this later line of research seems to dismiss techniques developed for multi-user toolkits such as MAUI[13] or other frameworks for rendering applications collaborative (i.e. CORK[11]). It follows, therefore that at present generating collaborative user interfaces for ubiquitous access remains ad hoc and highly programming-intensive task. As a result, it is argued that existing engineering practice is in search for a complete and coherent body of knowledge for collaborative UI development, thus leaving space for significant improvements towards this direction.

2. RELATED WORK: GROUPWARE TOOLKITS AND MODEL-BASED UI ENGI NEERING

2.1 Groupware toolkits

One prominent strand to developing collaborative UI's aims to exploit toolkit-based approaches to establish high-level support for either real-time distributed or single-display groupware. Although, each of these types of groupware applications pose different sets of challenges, they also share common ground. Specifically, they all make provisions for groupware widgets (e.g., telepointers, awareness widgets such multi-user scrollbars and radar views, etc), primitives for handling sessions, shared data models and communication-oriented details. The available literature and the documented experience with systems such as GroupKit[22],

EICS'09, July15–17, 2009, Pittsburgh, Pennsylvania, USA.

Collabrary[3], MID[1], MAUI, SDGToolkit[26] reveals both the complexity involved and the technical challenges to be addressed.

Despite the progress made thus far, it is also evident that the state of the art falls short of supporting, perhaps the greatest challenge a collaborative system should undertake, namely improved social awareness mechanisms between each member's activities in the context of a shared workspace. It is worth distinguishing social awareness from task awareness which is the level addressed by popular groupware widgets such as multiuser widgets, telepointers, radar views and annotations. Typically, these are supported through window or widget sharing. Nevertheless, social awareness is a more encompassing concept requiring understanding beyond the task level, thus raising new challenges at all levels. Despite the wide range of possible interpretations that the notion of social awareness has been related with (in respect to a particular cognitive domain, i.e.: sociology, cscw, etc.), there are some common elements upon we will argue. In the grounds of this conceptual commonality, social awareness should affect the UI's manifestation depicting the social relations/connections that exist among users of a certain collaborative session. Social awareness normally includes information about the presence and activities of people in a shared environment. The extent by which a UI answers to these questions acts as a catalyst determining the quality of teams' performance (Salas, Rozell, Driskell, & Mullen, 1999) and creativity. Characteristic representative of toolkits fostering social awareness though providing specialized fundamental building blocks/concrete computational underpinnings, is the "Café Construction Kit". Although current efforts towards this direction are in an infant stage.

2.1.1 The challenge of "Replication"

Depending on the potential requirements set by each project; different approaches had to be taken, as usual, leading in variations regarding their architectural structure. Main representatives of these variations include the replicated, the centralized as well as the hybrid architecture. Characteristic examples of the above architectures include Suite, Shared X, flexible JAMM[2], JASMINE[24], Bayou, NCSA Habanero[12], etc. One shortcoming is that these efforts provide rather custom than general solutions in the way that they don't provide distinct reusable components. Furthermore, they are tightly coupled towards the predefined problems they were initially programmed to solve while they don't provide the means, in terms of ease, for extending the predefined behavior of system's functional structural elements. This makes harder supporting the continuously increasing evolution of initially set up requirements.

By its nature the development of a collaborative system is a challenging task. This inherent difficulty is based on the complexity of the logical functionality they should perform, their computational manifestation as well as their obligation to support different features each time, for instance variant awareness indicators integrated in the UI widgets. Furthermore features could also include functional or not requirements such as extensibility, universal accessibility, adaptivity, adaptability, responsivity, etc. The previous are less or more well known issues in the CSCW community.

Furthermore, a very important aspect that toolkit based approaches ignore relates to the fact that they do not introduce software architectures and tools that support developers through the development life cycle. Approaches fostering/supporting software development through a development lifecycle have been introduced based on the model-based paradigm.

2.1.2 The challenge of "Synchronization"

The challenge of synchronizing information presented through suitable metaphors in distributed terminals is of major importance and concern in every collaborative system. Synchronization denotes the ability of a replicated object, through a suitable mechanism/architecture, to propagate a potential change to its state to the rest replicated objects in the context of a particular collaborative session so as to preserve data consistency.

Examples of different approaches include: the propagation of low level graphics instructions, i.e.: drawLine(x, y);, encountered in screen/application sharing approaches, the propagation of serialized events encountered in native to native UI synchronization or more recently the propagation of abstract notations, usually messages based on particular xml dialects, describing what was changed each time.

2.1.3 The challenge of "Floor Management"

The floor manager concerns with the functional part which is responsible for maintaining the order under which is given the permission to each participant to manipulate the replicated objects within the shared workspace. Indicative examples of different algorithms/policies for accessing the floor control include: Round Robin based, Role based, time slicing, etc. There have been proposed two main ways for implementing floor management architectures, namely: centralized and decentralized. In the first case, as architecture's name implies, there is a common/central point, usually in the server side, managing sequentially every floor control request. In the second case, each interactive object is associating, at the client-side, with a dedicated to it floor manager.

2.2 Model-based UI engineering

Model-based development as currently applied to user interfaces exploits device-independent markup languages and some sort of abstraction-based mechanisms to facilitate transformation of abstract interaction components to concrete widgets as implemented by a target platform.

Typically, the user interfaces built are single-user and non-collaborative. Furthermore, they seem dismissing techniques developed for multi-user toolkits such as MAUI or other frameworks for rendering applications collaborative (i.e. CORK). Furthermore, we should notice that until now Model-based UI generation was dealing with rather simple/simplistic problems which were, until now, far away from producing professional fully functional applications. This constitutes the manifestation of MDD of collaborative UIs much more challenging but on the other hand a potential positive result would be twofold in terms of success by proving the capacity of Model-based development of UIs, to support professional: single user as well as collaborative UIs.

Finally, another issue copes with the ability of rendering the same model, in terms of MVC, in different views to interactive applications of a particular type. This could be a key feature for virtual communities in the context of cross-organizational collaboration, acting as a boundary object.

3. BRIDGING ACROSS GROUPWARE TOOLKITS AND MODEL-BASED UI ENGINEERING

To bridge the gap between groupware and model-based UI engineering, the present work aims to exploit the best features of each tradition to devise a new process for building collaborative capabilities in multi-platform user interfaces. We intend to approach

this challenge by extending a popular model-based UI engineering framework, namely UsiXML[15], so as to support the required capabilities for groupware.

Following the model-based paradigm, UsiXML is based on the notion of models. Specifically it supports five types of models, namely: task, domain, context, abstract user interface (AUI) and concrete user interface (CUI) models. Each of these describes a UI in a different level of abstraction. In order to provide the desired collaborative orientation, UsiXML should be augmented properly so as to make provisions for managing multiple concurrent views on the same data which are suitably replicated and synchronized within a session. Furthermore, some sort of generic support for awareness should be provided so as to capture the social context within which collaborating peers engage in tasks. These features will need to appear either as extensions on the set of models already supported by UsiXML or as new models. Additionally, as client views may be distributed across different platforms it is important to address issues related to toolkit-level interoperability.

3.1 Extensions in the task & concepts analysis

Considering the various levels of a system's analysis, UsiXML adopts an extended version of CTT as the implementation layer for task analysis. Task analysis is useful for describing the various tasks to be carried out by users in interaction with an interactive system. So far, CTTs has been used mainly for modeling tasks of interactive single-user software applications. However, there are efforts (i.e., [18], [17]) reporting application of CTTs in a collaborative context. It seems that these works fail to account for user roles and multiple role-based views on the same collaborative task.

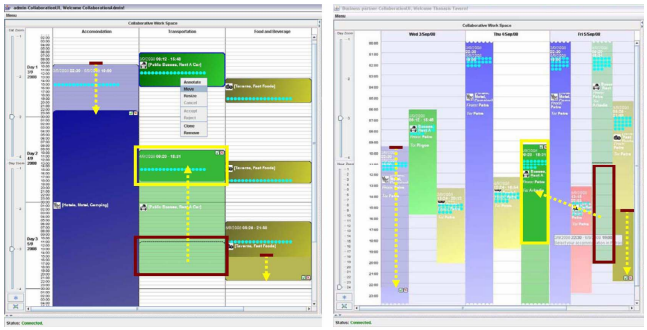

Figure 1: Replication of Role-based views

Our approach aims at fulfilling this gap. In particular we propose an improved task model, as an augmented version of the existing ones, merging the knowledge obtained from the current into a new one which would be taking care of user role differentiations (as in those depicted in Figure 1) and their effect in the whole process.

3.2 Extensions in abstract and concrete UI models

Providing participants with mutual awareness ([10] and [16]) of each other's activities is often seen as an important research and design goal (e.g. [6]; [25]). Awareness is provided to the potential users of a collaborative session through the UI. In UsiXML the models that are more closely to this type of details, are the Abstract and Concrete UI models. An example of these types of model, representing a specific UI, is shown below.

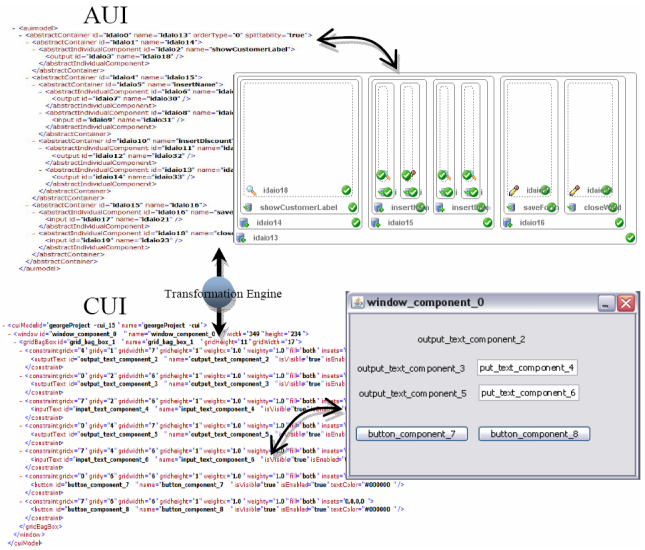

Figure 2: Graphic & Textual representation of AUI and CUI - models

These models should be augmented appropriately so as to capture the social aspects of collaborative applications (i.e., task, activity and social awareness) which are manifested through the user interface. Specifically, UsiXML's meta-model should be augmented so as to introduce an appropriate place where the UsiXML developer could define the desired distribution of events. Take for instance the case in which we would like to replicate a simple button. Let's suppose that given the appropriate tag (e.g. <element type=`button`>) a special UsiXML-compatible mechanism reading the appropriate UsiXML file could force multiple instantiation of the button through replication (distributed rendering) across all participating users' terminals. Furthermore another mechanism should be introduced so as to provide the means for defining the type of events (i.e.:mouseOverEvent, buttonSelectionEvent, etc.) that users would like to be propagated to each other. These directives could be injected directly to the CUI model or in another extra model, for example the "CUI_E" model as shown in Figure 3.

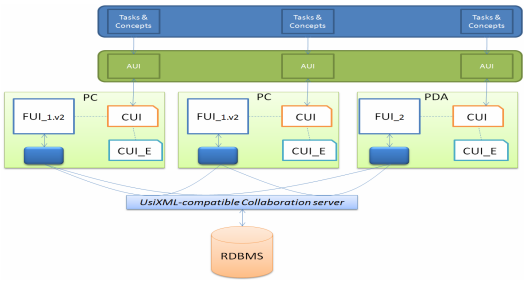

Figure 3: Architecture overview

Given the synchronous settings, a special mechanism should be present in each participant's side, so as to handle the capturing, distributing and repeating of local or remote events respectively. This functionality is depicted in Figure 3 as an empty blue box. This is in contrast with asynchronous collaborative systems where no care is required forcing runtime propagation-UI synchronization. Furthermore, except from defining the types of events that should be propagated there should be defined a new meta-language for the abstract definition of the potentially exchanged replicated events, taking care the possible context and platform variations.

3.2.1 Floor Management implications

Floor management, as a very important component met in almost every synchronous collaborative application, links with the extensions in abstract and concrete UI models in a manner implying several modifications to their conceptual design. Specifically, these models need to be augmented properly so as to provide UI designers with the means for defining which interactive elements, as parts or subsets of a particular UI hierarchy, would be under floor manager's supervision. This is very important so as to enable the manifestation of links among replicated objects and specific floor managers/policies.

We contemplate that the centralized architecture suits best in regard to the present context because of its simplicity and flexibility.

4. CONCLUSION & FUTURE WORK

The above provides only a non-exhaustive sketch of the current thinking and approach to address the designated challenges. Moreover, there are additional issues requiring further study before it is possible to devise a solution.

5. REFERENCES

1. Bederson, B., and Hourcade, J. (1999): *Architecture and implementation of a Java package for Multiple Input Devices (MID)*. HCIL Technical Report No. 99-08, May. http://www.cs.umd.edu/hcil
2. Begole, J., Rosson M. B., Clifford A. Shaffer: *Flexible collaboration transparency: supporting worker independence in replicated application – sharing systems*. TOCHI'1999, vol. 6, pp. 95-132.
3. Boyle, M. and Greenberg, S.: *GroupLab Collabrary: A Toolkit for Multimedia Groupware*. In J. Patterson (Ed.), CSCW'2002, Workshop on Network Services for Groupware.
4. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouil-lon, L., Vanderdonckt, J.,: *A Unifying Reference Framework for Multi-Target User Interfaces*. Interacting with Computers, Vol. 15, No. 3, June 2003, pp. 289-308.
5. Dewan, P., Choudhary, R.: *Coupling the User Interfaces of a Multiuser Program*. ACM Trans. TOCHI'95, vol. 2, pp.1-39.
6. Dourish, P., Bellotti, V.: *Awareness and Coordination in Shared Workspaces*. CSCW'92, pp. 107–114.
7. Florins, M., Vanderdonckt, J.: *Graceful Degradation of User Interfaces as a Design Method for Multiplatform Systems*. IUI'2004, pp. 140-147.
8. Garrido, J.L., Gea, M., Rodríguez, M.L.: *Requirements Enginnering in Cooperative Systems*. In Requirements Engineering for Sociotechnical Systems, Idea Group, Inc., pp. 226-244. 2005.
9. Guerrero, J., Lemaigre, Ch., Gonzalez Calleros, J.M., Vanderdonckt, J.: *Towards a Model-Based User Interface Development for Workflow Information Systems*. International Journal of Universal Computer Science, Vol. 14, No. 19, 2008, pp. 3236-3249.
10. Gutwin, C., Greenberg, S.: *Workspace Awareness for Groupware*. Human Factors in Computing Systems'96, pp. 208-209.
11. Isenhour, P., Rosson, M., B., Carrol M. J. (2001): *Supporting Interactive Collaboration on the Web with CORK*. Interacting With Computers, vol. 13, pp. 655-676.
12. Jackson, L., S., Grossman, E. (1999): *Integration of synchronous and asynchronous collaboration activities*. ACM Computing Surveys, vol.31.
13. Jason, H., Cutwin, C.: *The MAUI Toolkit: Groupware widgets for Group Awareness*. CSCW'05, vol. 13, pp. 539-571.
14. Lee, C., Helal, S., Lee W. *Universal Interactions with Smart Spaces*. IEEE Pervasing Computing, 2006, pp. 16-21.
15. Limbourg, Q., Vanderdonckt, J.: *UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence*, in Matera, M., Comai, S. (Eds.), "Engineering Advanced Web Applications", Rinton Press, pp. 325-338.
16. McDaniel, Susan E.: *Providing Awareness Information to Support Transitions in Remote Computer-Mediated Collaboration*. CHI'96.
17. Molina, A. I., Redondo, M. A., Ortega, M.,: *A Conceptual and Methodological Framework for Modeling Interactive Groupware Applications*. CRIWG' 2006, pp. 413-420.
18. Mori, G., Paterno, F., Santoro: *CTTE: Support for Developing and Analyzing Task models for interactive system design*. IEEE Transactions on S.E., 28(9), (2002) pp. 1-17.
19. Nichols, J., Myers, B.A., Higgins, M., Hughes, J., Harris, T.K., Rosenfeld, R., Pignol, M.: *Generating Remote Control Interfaces for Complex Appliances*. UIST'2002.
20. Penichet, V., Lozano, M., Gallud, J., Tesoriero, R.: *Analysis Models for User Interface Development in Collaborative Systems*. CADUI'08.
21. Penichet, Victor M. R.: *Task-Oriented and User-Centred Process Model for Developing Interfaces for Human-Computer-Human Environments*. PhD. University of Castilla-La Mancha.
22. Roseman, M., and Greenberg, S.: Building *Real-Time Groupware with GroupKit, a Groupware Toolkit*. ToCHI'96, pp. 66-106.
23. Schwartz, M., and Task Force on Bias-Free Language. *Guidelines for Bias-Free Writing*. Indiana University Press, Bloomington IN, 1995.
24. Shervin, S., Abdulmotaleb, S., Georganas, D. N., & Steinmetz, R., (2004)*: JASMINE: A Java Tool for Multimedia Collaboration on the Internet.* Multimedia Tools and Applications, vol. 19, pp. 5-28.
25. Tollmar, K., Sandor, O., Shomer, A.: *Supporting Social Awareness at Work - Design and Experience*. CSCW'96, pp. 298-307.
26. Tse, E. and Greenberg, S.: *Rapidly Prototyping Single Display Groupware through the SDGToolkit*. Proc. 5th Australasian UI Conference, Vol. 28 in the CRPIT Conferences in Research and Practice in Information Technology Series, Australian Computer Society Inc., pp. 101-110.
27. Wesley, Willett, Jeffrey, Heer, Maneesh Agrawala: *Scented Widgets: Improving Navigation Cues with Embedded Visualizations*. IEEE InfoVis'2007, vol. 13, pp. 1129-1136.
28. Giraldo,W., J., Molina A., I., Ortega, M., Collazos, C., A. : *CIAM: A methodology for the development of groupware user interfaces*. pp. 142-149. TAMODIA'2008.