Modeling and Design of Complex Cooperative Software

Max Mühlhäuser
University of Linz, Austria

Abstract

Recent technological advances have boosted multimedia and mobile computing. The application software market has seen an increasing interest in workflow and workgroup computing. These trends lead to a desire to build complex cooperative (mobile multimedia) application software. We propose a comprehensive software model and an interrelated set of five graphical view types for the modeling of such complex cooperative software. A middle-out design methodology is proposed for coordinated use of the five view types, starting with a scenario-based view type called work scenario. A first version of a corresponding multi-view editor was developed. The paper will concentrate on work scenario views and give an overview about the four other view types and the methodology.

1. Introduction

There have been many attempts to combat the software crisis with improved modeling and design techniques. We believe that such techniques are too much conceived as abstractions for (e.g., procedural or object-oriented) program structures, instead of reflecting customers' concerns. In the Items project discussed here, software development support is conceived around two kinds of customer concerns:

- organization-driven concerns: we claim that application software is more and more required to be *holistic* (i.e., to integrate formerly disjunct tools and tasks) and *work-centered* (based on activities rather than data schemes).
- technology-driven concerns: the trends toward *multimedia* and *distributed, even mobile computing* must be reflected in application development efforts.

In the remainder, we will present the graphical modeling and design approach of Items which reflects these concerns.

Holistic software design is the overall focus: 5 interrelated graphical "view types" are provided, ranging from enterprise-wide "organization models" to details of user interface organization and software configuration/mobility support.

Work-Centered design is supported based on a merge of workflow management and workgroup computing (CSCW) approaches; these two disciplines are not yet well integrated in the state of the art, cf. [8].

Multimedia is covered as to both (persistent *and* conversational) *multimedia* data (cf. [6]) and *multimodal* user interfaces (cf. [4]). According to good design practice, the choice among alternative media (e.g., audio-only vs. AV conference) is deferred as long as possible, and the use of alternative multimodal user interfaces (e.g., pen-based for notepad use, speech/mouse based for MPC use) is supported with a novel software design approach (not reported here in detail, cf. [2]).

Distributed and mobile computing support is based on a distributed object-oriented approach (extending [5] according to [9]) where organization-centered design is translated into a concrete configuration of software, documents, and user interfaces, all conceived as mobile objects, and mapped onto elements of the network topology which in turn contains mobile parts, too.

Figure 1 illustrates the relation of the four above-mentioned concerns onto the five view types in Items.

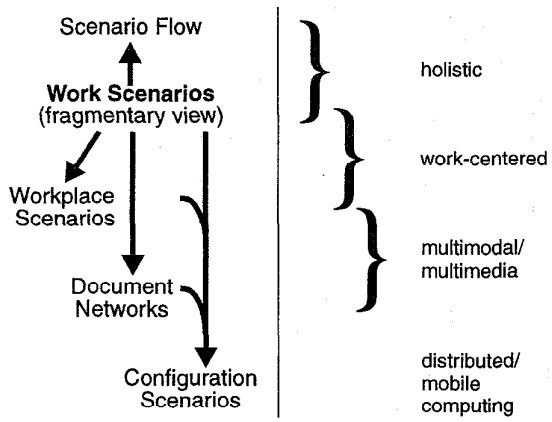

Figure 1. View types in Items

2. An overview of Items.

The development of Items has been governed by the attempt to apply *graphical* techniques extensively, supporting several interdependent *views* on a model rather than expressing everything in a single complex view. *Scenario-based* design is supported, i.e. the ability to express arbitrary fragmentary snapshots of an application as valid views of the model. A *middle-out* design approach was preferred, starting with scenario-based fragments. Intuitive design is supported, e.g., with notations for situations found to be common to

Proceedings of the 1st International Conference on Engineering of Complex Computer Systems (ICECCS'95)
0-8186-7123-8/95 $10.00 © 1995 IEEE

dynamic systems design like "there is an imprescriptible, dynamically changing number of components of type X - an arbitrary one of these can play the following special role: ...". The underlying graph concept supports Higraph-like hierarchical decomposition (cf. [3]). The fact that all view types support hierarchy will not be repeated each time below.

The five major view-types offered in Items will be described in the following (cf. fig. 1) in the order which corresponds to the proposed middle-out approach.

Work Scenarios: the designer is supposed to start modeling an application by specifying so-called work scenario graphs WSG. Every WSG specifies a way in which (usually several cooperating) users are linked to key application components and to one another. WSG are based on three *node* types (user, agent, archive) and six *link* types and describe incomplete scenarios of cooperative interaction among humans, software components, and persistent information. WSG will be described more in-depth in the next section.

Scenario Flow: in a scenario flow graph, the WSG are shrinked into a single node each. Such a node is then called "cooperation description unit" (CDU). Different scenarios (and thus, different CDU nodes in the scenario flow) may represent different kinds of "parts of" the overall system: different *fragments* of a large system, different (parallel or consecutive) *steps* in a business process of the organization, or different *situations* that may emerge based on different circumstances (e.g., conditional rules). In comparison with traditional approaches, steps and situations can be regarded as workflow steps and CSCW-like cooperations, respectively. Note, however, that they can be interleaved and layered at will. A scenario flow graph usually models a part of an organization in a work-centered way. It consists of an arbitrary-depth hierarchical decomposition of the holistic design into steps, situations, and fragments, all represented as nodes of type CDU. Links between CDU nodes model flows of activity. Three principal categories of flows exist: *control flow, role flow,* and *document flow.* All these are specified in part via annotated links between CDU nodes and in part via a set of rules to be observed at runtime. *Role flow* links specify relations between users or user groups of different scenarios; *document flow* links describe how persistent information is forwarded, accessed concurrently, etc.; *control flow* links exist in different variants and describe details of the transitions among situations and among steps.

Document Networks: based on the *archive* specifications which designers provide in the set of all WSG (cf. sect. 3), the relevant document types can be automatically extracted. These are then specified in more detail in document network graphs, one per document type. Document networks specify the internal document structure on the semantic level. They are modeled as graphs consisting of "units of information" in the sense of hypertext nodes. In contrast to the traditional hypertext concept, a document network may leave open the number of nodes in a sequence or in a "row" (a row denotes a bunch of links with the same inbound node and different outbound nodes, or vice versa; see [7] for the underlying concept). This way the logical document structures can be specified in a hypertext-conformant way, more precisely than with known document architectures, but still deferring decisions about concrete instance numbers (e.g., number of paragraphs and cross references) to runtime. The system-defined node types (*fixed:* presentation-only, for forms, video-clips, etc., *transient:* live, e.g., from a video conference, *virtual:* for navigation &editing rules, multimedia synchronization, etc., *pre-edited, blanc*) and link types refer to the document architecture; they must be complemented by user-defined semantic types and access rules to reflect the organizational model.

Workplace Scenarios: workplace scenarios (together with associated non-graphical information) emphasize workplace integration and *multimodal* user interfaces. Based on the *user* nodes specified in the set of all WSG (cf. sect. 3), all user interfaces (UIs to archives, agents, and other users, equal to the links to corresponding WSG nodes) that may be used by a user (role) are retrieved and assembled in a single so-called workplace (note that this workplace integration scope goes beyond usual UI design approaches). Starting from this initial outline, the UIs assembled in a workplace scenario are grouped and refined, using an elaborate software development methodology. It supports modality-independent UI design, deferring modality-dependent decisions to a late phase of software development, and encapsulating the core software such that it can switch to different modalities (speech, pen, gesture ...) at runtime. Cf.[2] for details.

Configuration Scenarios. This view-type is related to the distributed and mobile computing aspect of Items. It provides a mapping of the semantic application model (defined in the other views) onto software configurations with mobile objects and further onto a network which contains mobile devices.

First, the semantic application model, as given in the scenario flow graphs, is decomposed into independent parts: the CDUs in the scenario flow graphs which denote atomic *steps.* Atomic here means "not containing further *steps* but only *fragments* and *situations".* For each atomic step, the corresponding WSG are merged into one semantic configuration. The corresponding archives are decomposed into documents. All elements may then be further decomposed into so-called *shreds*, leading to, e.g., distributed user interfaces (*UI shreds*) which can be distributed between a palmtop and an Office PC, replicated data (*document shreds*) for "disconnected operations" in mobile computing, or *agent shreds.*

The resulting *software configuration scenario* is then extended to a full graph by mapping it onto a *logical hardware configuration scenario.* The latter consists of three node types: logical *networks*, (optionally mobile) *logical-nodes*, and mobile *node-shreds* (e.g., plug-in memory, Smartcards). As part of the mapping, *roamings* specify rules and paths of mobility for the mobile parts of both software (mobile objects) and logical hardware (logical-nodes, node-shreds).

Proceedings of the 1st International Conference on Engineering of Complex Computer Systems (ICECCS'95)
0-8186-7123-8/95 $10.00 © 1995 IEEE

Due to the lack of space, the remainder is restricted to a more detailed description of WSG. For details on scenario flows, workplace scenarios, and the document network base concept, interested readers may consult [1], [2], and [7].

3. Work Scenario Graphs

WSG are much inspired by whiteboard sketches as typically used in very early design for capturing ideas. Scenarios have been identified as an appropriate means for requirements engineering, particularly in the context of human-computer and human-human interaction, since they depict both situations and dynamics, and since they depict situations in a larger context which includes, e.g., resources and intentions, cf. [9]. WSG are composed of only three types of nodes:

Agent nodes describe "substantial" self-contained software components. These may be realized as distributed, multi-threaded software. Agents represent autonomous functional entities from the point of view of the organization-centered design; not every piece of application software is part of an agent, it may alternatively "belong" to a user or archive node or even – e.g., as configuration control software – to a link. As an overall design rule, the design models the parts and relations which are of interest to the user.

Archive nodes depict collections of documents, stored in single or replicated files or databases: they model persistent information. Access tools for archives like editors, browsers, database interfaces, or TP monitors are assumed to be part of an archive. As a counter-example, an autonomous agent for information filtering would be modeled as an agent.

User nodes represent user access points rather than physical users or self-contained user interfaces. They may be viewed as a user´s end-point of a connection to either another user node or an agent or an archive. Both by filling out a specific "role" and by performing a specific task, a user may use several user access points simultaneously. It is the task of a workplace-integrating user interface (cf. sect. 2) to offer and coordinate a selection of user access points in a timely fashion, according to the tasks and roles which the user actually fulfils.

WSG nodes are linked together via undirected typed point-to-point or multipoint connections. Link types are fully determined by the types of nodes they connect; a link may connect either nodes of one type or nodes of two different types; accordingly, there are six possible link types:

Three link types denote the possible connections of users to other users (*converse* links, modeling synchronous or asynchronous conversation), agents (*use* links), and archives (*access* links, modeling document editing and information browsing). The three other possible link types are called *interact* (agent-agent), *interpret* (agent-archive), and *update* (archive-archive).

Links with equal participation of all three types of nodes are inhibited: instead, an attempt of the designer to draw such a link is interactively turned into one of the two more detailed link types which exist: a) links with an associated archive in the center, indicating that the "transient" communication on that link is to be recorded and stored (cf. fig. 3); b) link with an associated agent in the center, indicating that ("more than normal") sophisticated configuration and management is to supported.

The instantiation of WSG elements can be determined by adding a corresponding symbol. Three possibilities exist: a "0" denotes a node as optional, i.e. it may or may not exist in a concrete instantiation of the WSG; "1" nodes are mandatory, and "*" nodes may be instantiated arbitrarily often, the number of instances may even change at run time.

Of course, much of the "meaning" of a graphical design stems from the mnemonics associated (here: from the node identifiers). Unlike programming languages, WSG allow multiple use of identifiers within the same scope. E.g., one may want to depict that any single one within a set of nodes with same functionality (denoted by a single "*"-labelled node) may take on a particular role: this fact can be denoted by drawing another node with the same identifier but a "1"-label (cf. the example below). There exist even more possibilities to add information to a WSG, for example, about the concrete meshing of nodes associated with a multi-party link. Such further levels of detail will be omitted for the sake of space.

Design example: in the following, the basic concepts of WSG design will be illustrated through a partial design of a software system for distributed electronic meeting assistance.

In an early stage of design, the WSG designer may want to specify just the following facts: in an electronic meeting, the roles *moderator* and *participant* must be distinguished; the basic archive to be considered contains the electronic *handouts;* users must be able to communicate and to access the handouts. The resulting basic WSG is depicted in figure 1.

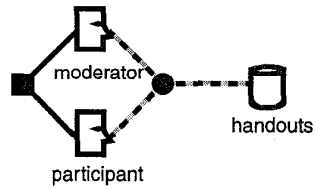

Figure 2. Initial WSG

In our sample design procedure, the initial WSG is altered, adding design decisions and introducing both a change in the topology and additional labels: electronic meeting assistance is found to require substantial software support by an autonomous component which dynamically manages the topic list, computer-supports discussions and decisions for each topic, and organizes time management, all in cooperation with the moderator. Therefore, an agent called EMA (*electronic meeting agent*) is introduced; due to EMA control, direct user

Proceedings of the 1st International Conference on Engineering of Complex Computer Systems (ICECCS'95)
0-8186-7123-8/95 $10.00 © 1995 IEEE

access to the handouts is excluded. An arbitrary participant is appointed for taking the minutes (cf. "1"-notation below).

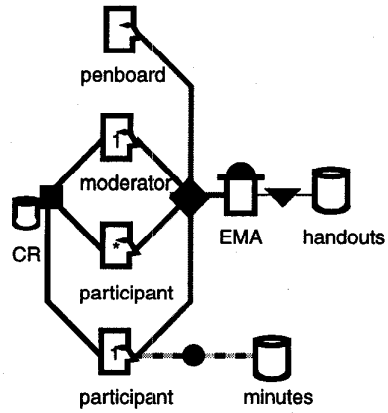

Figure 3. More detailed WSG

The resulting figure 3 shows two further details: a) All conversation is recorded in an archive called "conversation recording (CR)"; note that the decision about synkaryon and media used (e.g., audio-video conferencing or electronic mail) is not taken here, so that CR may represent, e.g., a folder for email "carbon copies" or an audio-video store. b) The EMA supports interaction via an interactive electronic whiteboard (node "penboard"). Such penboards are optionally available at each physical location which participates in the electronic meeting; activities which normally occur at a traditional whiteboard can thus be propagated to all locations involved in the meeting.

At this point in the design, the designer may want to introduce a second *fragment* in order to limit the complexity. This second fragment may express the fact that a so-called demo-agent allows to display complex instructional material on penboards during an electronic meeting. The demo_agent is software-controlled by the EMA and human-controlled by a single yet arbitrary participant. The separate fragment is created since on one hand, it introduces a new node and link; on the other hand, parts of the first fragment are not relevant in its context.

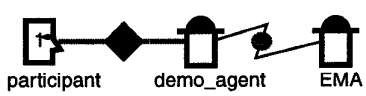

Figure 4. Second fragment within same design

Next, the designer may want to foresee the possibility to instantaneously enter a phase of formal negotiations at any time during a meeting. The participants would then take on new roles in a negotiation process as shown below (details of these roles are not important here). The EMA monitors the negotiation based on rules stored in a dedicated archive. Since the negociation context may be entered and left at will during a meeting (e.g., based on a user interface action), it is neither a parallel nor a subsequent step. This is a typical example for a separate *situation.* To summarize, figures 1 and 2 illustrate fragments of situation no. 1, figure 3 illustrates situation no. 2.

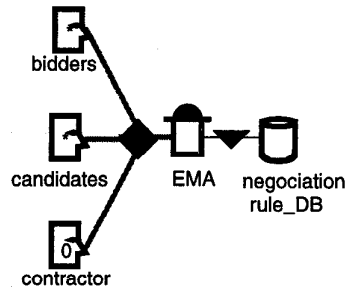

Figure 5. Second situation within same design

Altogether, these WSG could be combined into a step "electronic meeting", to be followed, e.g., by a different step called "minute-writing" (omitted here for the sake of space).

4. Summary, Status, Acknowledgments

We have exemplified one out of five view types for the design of complex cooperative (multimedia mobile) application software. We are currently gaining first experiences with a corresponding multi-view Items editor (not all views are implemented; in addition to the above, the use of "personalized" icons for different node identifiers is supported). The author would like to thank various Items project members and contributors from the Universities of Linz and Karlsruhe and from Eurecom (Sophia Antipolis, France), most notably O. Frick, HW. Gellersen, P. Mammay, and B. Merialdo.

5. References

[1] O. Frick, M. Mühlhäuser, H. Gellersen, Developing Cooperative Media-Integrated Software, in *Innovationen bei Rechnen- und Kommunikationssystemen*, B. Wolfinger (Ed.), Springer Berlin etc., 1994, pp. 227-234.

[2] H. Gellersen, Support of User Interface Design Aspects in a Framework for Distributed Cooperative Applications, in *Software Engineering and Human Computer Interaction,* R. Taylor (Ed.), Springer LNCS, 1994.

[3] D. Harel, *On Visual Formalisms.* CACM 31, 1988, pp. 514-530.

[4] J. Johnson, Selectors: Going Beyond User-Interface Widgets, in *Proc. CHI '92*, Monterey, CA, 5/92, pp. 273–279.

[5] M. Mühlhäuser, L. Heuser, W. Gerteis, DOCASE - A Methodic Approach to Distributed Object-Oriented Programming. *CACM* 36(9), 1993, pp. 127-138.

[6] *Multimedia in the Workplace.* CACM 36 (1), 1993.

[7] M. Richartz, M. Mühlhäuser, PreScripts: A Typing Approach to the Construction and Traversal of Hypertext, in *Educational Multimedia and Hypermedia,* H. Maurer (Ed.), AACE Charlottes-ville VA, 1993, pp. 436-443.

[8] T. Schäl. B. Zeller. *Supporting Cooperative Processes with Workflow Management Technology.,* Tutorial at ECSCW '93, Milano, Italy, Sept. 1993.

[9] Weiser M.: Some Computer Science Issues in Ubiquitous Computing, *CACM 36* (7); 1993, pp. 75–85.

Proceedings of the 1st International Conference on Engineering of Complex Computer Systems (ICECCS'95)
0-8186-7123-8/95 $10.00 © 1995 IEEE