Designing Cooperative Media-Integrated Mobile Applications

Max Mühlhäuser, Bernard Merialdo*,
Hans-Werner Gellersen, Oliver Frick, and Stefan Gessler

University of Karlsruhe, Institute for Telematics,
POB 6980, D-76128 Karlsruhe,
{max|hwg|frick|gessler}@tk.telematik.informatik.uni-karlsruhe.de

**: Institut Eurecom., POB 193*
F-06904 Sophia Antipolis Cedex, merialdo@eurecom.fr

Abstract: We believe that the trend towards cooperative applications is interwoven with trends towards both multimedia and mobile computing and with the trend towards integrated software solutions. This means that the design and development of cooperative applications should be treated in a larger context: the design and development of "customized enterprise-wide cooperative media-integrated (i.e. multimedia and multimodal) mobile applications" — *advanced cooperative applications* for short. In our paper, we will motivate the importance of such a holistic approach and introduce a design framework for advanced cooperative applications, called Items. This design framework draws from the ideas of scenario-based design and offers five interrelated view types for describing an advanced cooperative application at different levels of detail and from different perspectives. The cooperation-related view types will be emphasized in the paper. Items represents work in progress.

1 Motivation

Synergy: We view cooperative software systems as an attempt to promote the synergy of humans at, e.g., a working environment (enterprise). Such synergetic work is investigated today in two traditionally disjunct research domains which converge only slowly: *workflow computing* and *workgroup computing*.[1] Workflow computing addresses *rather* large groups / asynchronous cooperation / strict decoupling of individual activities, whereas workgroup computing addresses *rather* small groups / synchronous cooperation / closer coupling. Typical application examples are travel claim processing and joint editing, respectively. The characterization raises evidence that both fields represent only two extremes in a continuous spectrum of synergetic group work.

“Serious” synergy via software can only be effective if the software for group support and the software for individual daily work (TP, MIS, email software etc.) are closely coupled. This corresponds to an actual trend towards integrated software solutions reflected in terms like office automation, computer-integrated manufacturing, etc. (this might also be coined as “synergy *of* software”). Consequently, cooperative applications should be customized towards enterprise-integrated solutions, rather than be out-of-the-box groupware. This leads to the demand for *general* software technology for complex software (enterprise-wide, distributed) which *includes* cooperative-work aspects (which, in turn, spans the range from workgroup to workflow computing).

Ubiquity: today, individual (daily-work) software is typically desktop-based. To a certain extent, cooperative software can also be desktop-based: cooperation is in this case based on *mobile [software] objects* which mediate between group members (cf. desktop conferencing software and cooperating intelligent software agents). To a certain extent, however, cooperative software involves *mobile users* which, e.g., walk

1. The meaning of these terms is in transition: a year ago, one would have written “workflow management and CSCW”; note also that most of todays “workgroup” software on the market is not really “groupware”; CSCW is viewed more and more as the embracing term

around or travel in order to meet other humans; these mobile users may or may not carry *mobile devices* (flash memory, notebooks, …). In any case, the fact that their cooperation is software-based makes them depend on computers heavily and at virtually any place. In other words: the demand for (some degree of) ubiquitous computing increases heavily in the context cooperative software, and it has to be founded on support for mobile (software) objects, mobile users, and mobile devices.

Modality: multimedia plays a very special role as the linking element between "synergy" and "ubiquity". Obviously, multimedia helps to support remote cooperation, e.g., with audiovideo conferences; we even expect that multimedia data will make a transition from being used in rather specialized applications to being used in virtually all areas of normal enterprise computing: we coin this as the transition to media-integrated applications. From a modeling / design point of view, we see a high demand for two particular features in multimedia application development: *media transparency* and *orthogonal media*. In terms of modeling, it should be possible to abstract from the actual media used, e.g., for a conversation between several cooperating users; the choice between, e.g., audiovideo, audio-only, or mirrored keyboard input should be made at runtime rather than at design-time (media transparency). This feature leads directly to the requirement that different media may be used interchangeably (orthogonal media).

However, if audio and video shall be used in exchange of "well-understood" media like text and structured data, then all media must be semantically understood by the software system; we coin this feature as *modality.* As an example, imagine a "telemeeting" software which understands that "meetings" comprise "topics" which in turn comprise subparts like "introduction", "discussion" and optionally "decision". Imagine that different conversation media shall be allowed (media transparency, see above) and that the conversations during a meeting shall be recorded (on digital disk). If the reporter of a meeting wants to complete the minutes, s/he may want to access the conversation related to a topic discussion. In order to do this easily, every recording must be semantically structured into named topics and further into introductions, discussions, and decisions (and all this independent of the actual media used!). In this example, sophisticated speech or video recognition is not even required: given sophisticated telemeeting software, the topics and their subparts are treated with software support during a meeting. This means that the telemeeting software knows exactly at which time which topics and subparts are treated. Therefore, a simple time-code mechanism can be used to identify, e.g., portions of audio and video recordings that correspond to a topic/subpart.

Today, the term modality is mainly used in the context of human-computer interfaces (HCI); our view harmonizes multimedia and HCI-based multimodality in the common area coined as modality. A multimodal HCI interprets several input channels to the computer in order to recognize a user's input (commands). This HCI-related meaning of modality is important in the ubiquity context cited above: mobile cooperating users tend to use a broad range of "terminals" including PDAs, notebooks, PCs, multimedia workstations, liveboards etc. Such devices tend to exploit the input media (modalities) which suit best, such as pens, speech, and keyboards. If ubiquitous access to advanced cooperative applications is to be granted, such applications must be accessible from the whole range of such "terminals", i.e. with varying modalities. But then, software technology ought to support a modality-independent layer of HCI design *and* an adaption layer in which modality-specific aspects can be designed.

The three interdependent aspects of software technology for advanced (customized enterprise-integrating, mobile, media-integrated) cooperative applications are summarized below: enterprise-integration is reflected in aspect synergy, mobility in aspect ubiquity, and media-integration in aspect modality. The latter aspect is a kind of link between the other two.

$$advanced \left\{ \begin{matrix} enterprise-integrating \\ media-integrated \\ mobile \end{matrix} \right\} applications \leftrightarrow \left\{ \begin{matrix} SYNERGY \\ MODALITY \\ UBIQUITY \end{matrix} \right\}$$

2 Existing Approaches

For the sake of space, we will present a rather selective state of the art review.

Synergy: most existing *workgroup computing* approaches represent out-of-the-box tools. Among the counter-examples (support for customized solutions), the prominent ones include GroupKit [14] and GroupIE [15]. The GroupKit toolkit supports the development of conferencing applications. Its abstractions remain on a very low level, however, and lack notions for coordination support (e.g., floor control) and for more customized concepts than just “group” and “participant”. GroupIE offers more elaborate means for modeling workgroup computing solutions. Its seamless integration with the Smalltalk programming language fosters the development of fully “cooperation-aware” applications. GroupIE includes higher-level descriptive languages but remains entirely on the implementation level, i.e. does not provide design support.

In the *workflow computing* area, most approaches support customized solutions at first sight (see [17] for a comparative overview). Workflows are decomposed into sequences of isolated activities each of which must be distilled into a simple user-level operation like “open editor with file travel_claim.doc”; more complex or even cooperative “steps” are hardly supported [17].

To summarize, both workflow and workgroup computing solutions can be developed today, but there are hardly any approaches for covering the full spectrum between these two extremes, nor approaches to tightly integrate with all kinds of enterprise computing, let-alone concepts to model modality and ubiquity aspects at the same time.

Modality: *distributed multimedia* application development support is still mostly a research topic (cf. “MediaSpace” toolkits [9], modeling / implementation concepts such as HeiMAT [16], integrated development / support systems such as Mode [1], and many others). As one of the few industry standards, the IMA (Interactive Multimedia Association, including IBM, HP and SunSoft) has proposed a technology for

Multimedia System Services in a distributed environment [4]. In this technology for instance, real devices are controlled by objects called virtual devices which contain information about the actual capabilities of the real device. The description of virtual devices are supposed to be standardized across platforms. Virtual devices can be connected together under the control of a virtual connection object. In particular, this object performs format negotiation about the data being transmitted from one device to the other. The management of these distributed objects follows standardized (CORBA) technology. IMA and all other approaches handle explicit media types early on in the software development process, i.e. they do not support sufficient media transparency.

Approaches to media perception have greatly advanced in the past years and lead to fairly general-purpose concepts (cf. wordspotting and face-recognition technology), representing an important step towards orthogonal media and modality; however, these approaches have not yet made their way into the application development frameworks mentioned.

Multimodal HCI development is poorly support in present UI development approaches. Existing methods are paradigm-bound; they impose metaphors, modalities (usually mouse/keyboard-gesture and graphics display), media, and dialogue structure. Most design decisions are not supported explicitly but hidden in application code and can not easily be revisited. This results in UIs difficult to port and extend and difficult to adapt to, e.g., varying devices and user preferences. In particular, dialogue semantics are usually not kept separate from domain-related semantics. Only few UIMS support interaction semantics explicitly [7]. Cooperation support is usually missing in UIMS work [11], except for dedicated solutions [9]. Workplace-integration (for disjunct activities) does not draw from “workflow knowledge” [13] [12]. Support for multimodality is in an early stage only [7].

To summarize, current approaches towards distributed multimedia application modeling and development support neither a sufficient degree of media transparency nor do they address the aspects of orthogonal, interchangeable media. Support for “media spaces” combines cooperation and multimedia aspects, but leads to very specific kinds of applications. HCI design is kept as an issue separate from other multimedia aspects and from synergy and ubiquity aspects; it does not foster media-transparency.

Ubiquity: Ideally, ubiquitous computing can be reached either by offering accessible *on-site* computers virtually “everywhere” plus full home environment support (provision of the usual working environment known to the user) or by offering very powerful *portable* computers to every user plus full connectivity to both the home environment and the local non-portable resources (such as high-quality printers) from “everywhere”. Today, a fairly restricted level of the second solution is state of the art, at its best combined with some home environment support which maps logical resources of the home environment (e.g., “Postscript Printer”) to the actual environment. Corresponding research usually assumes low connectivity of users outside the office (“low” with respect to both bandwidth and connection establishment / sustain probability) and is usually categorized as “mobile computing support”.

As to application support, the main focus in mobile computing research is on what we would call *mobility transparency:* the operating system supports mobile computing with no need for the application software to be changed. Such support is mainly focussed on so-called *disconnected operations* [6] [8] [10] which are embraced by the *hoarding* of necessary source files during high connectivity (e.g., based on manual user-interception using the briefcase metaphor) and by the *reintegration* of multiple copies of files or databases after reconnection (e.g., based on database reconciliation techniques). Studies have shown, however, that knowledge about application semantics can greatly reduce the number of unresolvable conflicts at reintegration time. This goal is only feasible if application design integrates a model of mobility (mobility-awareness).

Even more drastically, [18] argues that advanced ubiquitous computing requires a revision of many areas which were supposed to be well resolved, such as HCI modularization (cf. the network interface and protocol of X window™). These issues can only be resolved if mobility (or rather ubiquity in the above-defined sense) is modeled along with aspects of synergy and modality. Obviously, the “mobility model” in search has to describe the movements of users, devices, and software objects in the sense of a “configuration model” (much like the configuration support offered in distributed programming approaches, but with a focus on mobility).

Up to now, the mobile computing community has discussed configuration models mainly for the design and evaluation of issues in the lower layers of the communication architecture (physical, media access, network, etc.), and *not* in the context of application support.

To summarize, mobile software objects and mobile users/devices are not regarded in a common context today. Support for mobile users and devices is investigated in the “mobile computing” community, but usual solutions provide operating system level support for mobility-transparent applications which is often insufficient. Rather, software technology should support the development of mobility-*aware* applications; such support should include mobility-focussed configuration models and ways for communicating application semantics to, e.g., the “disconnected operation” support.

3 An overview of Items.

Project Items aims at software technology for advanced cooperative applications which include media-integration (multimedia, multimodality) and mobile computing and which represent enterprise-wide software solutions. Such advanced cooperative applications are by nature complex and include aspects which are traditionally addressed in dedicated software development approaches like the ones described in section 2. Items, in contrast, addresses these aspects as part of a *general* software technology. In the remainder, we will concentrate on the *modeling and design* of such advanced cooperative applications as emphasized in the first phase of project Items.

Our primary goal was the provision of a means for expressing designs such that users and software engineers could understand and discuss them. Given the high complexity of advanced cooperative applications, this was found even more important than the ability to generate code or to apply formal validation methods on a model (these are of course *also* goals of the project). According to this primary goal, the following principles have governed the design of our modeling (/design) framework:

- support synergy, modality, and ubiquity in a single, common framework;
- apply *graph-based* techniques extensively;
- support several interdependent *views* on a model rather than express everything in a single complex view; different *types* of views should be possible;
- support *scenario-based* design, i.e. the ability to express arbitrary fragmentary snapshots of an application as valid views of the model;
- promote a *middle-out* design approach as the preferred method, without forcing the designer to follow a rigid modeling technique; rather the modeling system, with its support for keeping multiple views consistent, should leave enough freedom to choose at any time – within limits – among several possible views to elaborate further.
- enable intuitive design in many ways: by permitting different alternative ways to describe a certain fact; by supporting easy-to-use notations for situations common to dynamic systems like “there is an unprescribed, dynamically changing number of components of type X - an arbitrary one of these can play the following special role: …”; by supporting hierarchical hyperlinked graphs as introduced by Harel [5]; by allowing rough and refined ways of graphical specification; etc.

The five view types. The major five view-types offered in Items will be described in the following, as depicted in figure 1. The sequence in which they are explained below corresponds to the proposed middle-out approach in which the views can be specified (remember that designers can switch between views at will, provided the

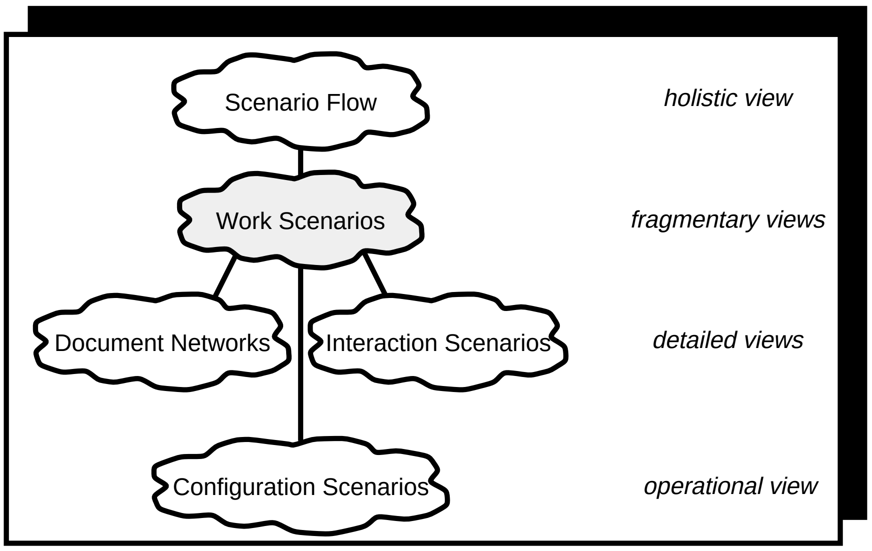

FIGURE 1. View types in the Items modeling framework

necessary minimal input from one view to another has been furnished).

Work Scenarios. For a middle-out design, the designer is supposed to start modeling an application by graphically specifying so-called work scenarios. Each such work scenario specifies a way in which (usually several) users are linked to major application components and to one another. Scenario graphs are based on three types of *nodes.* Nodes of type "user" model the access of a user to the application (the user interface, in the largest sense) – within the given scenario, in a specific role, and abstracting from a specific person via this role. Nodes of type "archive" represent data repositories such as files or (maybe replicated, maybe globally distributed) databases. Archives provide access to (a limited set of kinds of) documents, were the notion of a document has a wide scope, see below. The third kind of node is called "agent". It denotes self-contained (maybe distributed) substantial non-obvious software components. Counter-Examples i.e. "obvious" or "non-substantial" components – which would *not* be modeled as separate agents – comprise editors and browsing components for archives and documents, control components for connections (such as conference managing modules for videoconferences), and everything that could be called a part of a "user" node.

Several interconnected nodes of types user, agent, and archive form a scenario.

Scenario Flow: scenario flows and work scenarios together, along with the below-mentioned document networks, describe the *synergy* aspect of an application. In a scenario flow, the above-mentioned work scenarios are shrinked into a single node each. Such a node is then called “cooperation description unit” (CDU). Two scenarios (and two CDU nodes in the scenario flow) may be related in very different ways and degrees. First, they may indeed be two *fragments* of an application which run at the same time and under the same circumstances. Secondly, they may represent two different cooperation *situations* of a group of people, interrelated basically by events (a telemeeting topic with or without an expert to be called upon, a request for voting suddenly raised, etc.). Thirdly, they may represent two subsequent or parallel *steps* in a workflow-like setup. It is the task of the scenario flow design to identify scenarios as fragments, situations, or steps, to compose and interrelate them, and to identify the flow of resources, roles, documents etc. via different link types (cf. Section 5).

In comparison with traditional approaches, steps and situations can be regarded as similar to the building blocks of workflow and workgroup computing, respectively. Note, however, that we do *not* prescribe a hierarchical order of these, so that, e.g., “lower-level” (workgroup type) situation-based layers of a scenario flow would have to be subordinate to “top-level” (workflow type) step-based views. They can be interleaved at will, just not in the same meta-node. Section 5 will also give an idea about how steps and situations can be used to cover the seamless spectrum of cooperation types.

Document Networks. Following the proposed design approach, relevant document types are first identified in work scenarios and then specified in more detail in the views called “document network”, defining one such view per document type. (Another approach may be to center an application design around the relevant documents and thus to start the design from the document networks.) Document networks specify the internal structure of a document on the semantic level (the detailed document layout is not part of the Items model). Document types are modeled as graphs consisting of "units of information" in the sense of hypertext nodes. In contrast to a concrete hypertext, a document network allows to leave open the number of nodes in a sequence or in a "row" (a row denotes similar nodes which are all linked to the same preceding node). Thus e.g., document structures with an open number of paragraphs and cross references can be specified. Predefined node types in a document network comprise "fixed" (presentation-only, for forms, multimedia-information, etc.), "pre-edited", "blanc", "transient" (furnished live by an application or by a conversation such as videoconference), and "virtual" (for navigation / editing rules, multimedia synchronization, etc.).

Interaction Scenarios: interaction scenarios, together with associated non-graphical information and with information from the work scenarios, provide for the *modality* aspect (cf. [3]). The designer identifies one or more "user" nodes to be considered in a common HCI. If several nodes are considered, they are supposed to be related to the same physical user at a time (note that this "workplace integration" approach goes beyond the scope of todays user interface design approaches). From the work scenarios, all connections of these "user" nodes to other nodes (of type agent, archive, or user) are retrieved. Thus the coarse "outline" of the workplace is already derived automatically. Based on this coarse outline, the interaction scenarios are used to define more details about the HCI for each connection (available commands and parameters, their organization in contexts etc.) as well as for the overall workplace-wide HCI design. In a first phase, this whole design is media-transparent and modality-transparent. Only in a second phase, modality specific information is added.

Configuration Scenarios. The fifth view-type is related to the ubiquity aspect of Items. It provides a mapping of the "logical" application configuration (as defined in the other four views) onto "operational" physical configurations with mobile objects, users, and devices. The major design steps and building blocks are briefly sketched here.

The logical configuration is based on the lowest level of step-type CDUs in the scenario flow. For each such step, the transitive closure of situations and the join of fragments is built. The resulting set of users, agents, archives and interconnections is computed and graphically depicted. Non-relevant parts, i.e. non-mobile nodes, interconnected only to other non-mobile nodes (and users), can then be shrinked to blackboxes. For the remainder, “fragments” may be defined: user fragments define distributed user interfaces which can, e.g., be distributed between PDAs and connected OfficePCs; agent fragments define distributed applications; archive fragments define either replicated or distributed information, e.g., for disconnected operations. With these design steps (merge of all CDUs which belong to a step, expansion into the merge of all related work scenarios, shading-out of parts which are irrelevant for mobility, definition of fragments), a logical configuration (mobility-compliant view) of the application step is created.

Next, an operational configuration is created by specifying relevant physical devices and device fragments (PCMCIA cards, etc.), networks (as relevant from a mobility point of view) and their interconnections. Afterwards, logical configurations are mapped onto operational ones and complemented with information about the required mobility support (“roaming domains” and “roam paths” of the mobile users, “Quality-of-Service Specifications” such as necessary security and consistency/reconciliation support, etc.).

Following this overview of the five basic view types of Items, we will restrict ourselves to a more detailed description of the first two (most cooperation-related) view types for the sake of space.

4 Work scenarios

Scenarios have been identified as an appropriate means for requirements engineering, particularly in the context of human-computer interaction and human-human cooperation, 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].

Scenarios tend to be comprehensive in the sense that they include people involved, flow of information, tools, etc. However, they do *not* intend to describe an application "completely" at any level of abstraction; rather, a set of scenarios is supposed to be needed to reach completeness.

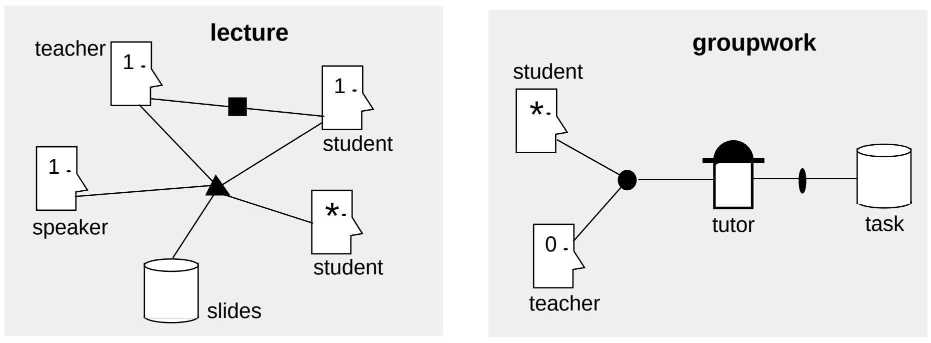

FIGURE 2. work scenarios "teleteaching"

In order to give an idea about work scenarios in Items beyond what was explained in section 3, we will use two simple example work scenarios from the teleteaching domain, as depicted in fig. 3. These two work scenarios may be considered two steps of a workflow-like scenario flow. The graphical symbols for the three node types seem to be obvious ("agents" carry a butler's bowler hat). In work scenario "lecture", several students ("*" notation) and one teacher ("1" notation) are connected to the lecture notes ("slides"). At any time, a single student ("1" notation for a node with the same identifier) may be linked to the teacher directly ("conversation link"). Another explicit user role "speaker" is defined, supposed to be taken on by either the teacher or a student. This shows that nodes of type "user" denote either organizational roles or activity-related roles; the latter are related to a specific interaction context (speaker, listener). Most often, organizational roles imply a certain behavior i.a. activity-related role (student[*]=listener).

Scenario "groupwork" models the interaction between a group of students and an intelligent tutoring system “tutor” which guides the group of students based on a high-level task description “task” which includes descriptions of possible misconceptions and the student models. The teacher may connect to the group in order to respond to particular questions. Since the teacher is optional, it is depicted using the “0” notation.

The above-mentioned scenarios illustrated several kinds of connections between nodes, classified via the node-types they connect: user-user (conversion), user-archive (EditBrowse), user-agent (dialog), agent-archive (perceive); agent-agent connections (messaging) exist in addition. Note that the conversion connection exhibits media transparency.

The scenarios described can be refined, e.g., in order to give more precise information about the multiplicity of links, micro-coordination (access and operation policies, etc.), macro-coordination (constraints) etc. Such detailed information provides further interweaving with the other view types (macro-coordination ↔ scenario flow, cooperation policies ↔ interaction scenarios, etc.).

5 Scenario Flows and Synergy

We recall the reasons for defining different work scenarios within a scenario flow:

- a complex scenario may be separated into *fragments* that happen at the same time and under the same circumstances.
- different work scenarios may describe different application *situations*, where all situations are rather similar (e.g., share a number of common archive, user, or agent nodes) but happen at different times or circumstances. The transition between two situations is, for example, triggered by an event and results in a slightly but structurally different scenario.
- a set of work scenarios may be fairly disjunct and describes different subsequent or parallel *steps* that make up a workflow-type application.

In order to connect a set of work scenarios to a scenario flow, all work scenarios are shrunk to their single node representation called “cooperation description unit” (CDU). Connections between work scenarios are established by means of *flows* which graphically depict relations between nodes (on the work scenario level) and CDUs (on the scenario flow levels), respectively – cf. figure 3-5. Just as the node types make up the “design center” for work scenarios, flow types make up the design center for scenario flows. Key flow types include: *role flow* links which symbolize relations between users or user groups of different scenarios; *document flow* links which define relations between archives; *control flow* links which define relations between agents; *identical* links which are used to define nodes within application fragments as being identical (cf. figure 3).

Instead of describing the three kinds of scenario occurrences (fragment, situation, step) and the different flow types in full detail, we will just give three examples of small scenario flows below for the sake of space, continuing with the teleteaching example from figure 2.

Figure 3 splits the “lecture” work scenario of figure 2 into two parts (note the freedom in the design to depict the application either way); the “speaker” node is omitted for the sake of simplicity. The first fragment shows that the teacher presents lecture notes (slides) to all students, the second one shows that the teacher may in addition (!) open a private conversation if a student has queued a question in the slides archive. As the teacher as well as the slides are identical in both fragments, “identical” links are drawn accordingly. The figure shows how scenario flow designers add information to the work scenario level, it also shows the depiction on the scenario flow level itself (CDU-based). On that level, the notation “identical: add” indicates that private conversations represent an optional add-on; the declarations in square brackets show *who* has the right to initiate a private conversation (teacher), *which conditions* must be satisfied (a question must be stored), and *how* the transition is effected (by calling operation S_private which, e.g., identifies the student to talk to).

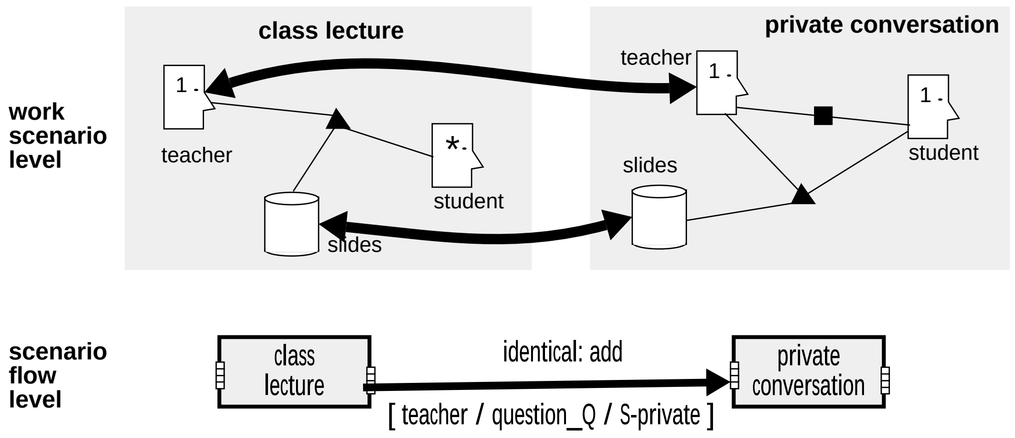

FIGURE 3. Scenario flow of work scenario fragments

Figure 4 depicts a scenario flow with two *situations:* we suppose that the teacher may, at any time during a lecture, suddenly decide to switch to an "audit" mode where a selected student has to rephrase a subject. This second situation is supposed to have a threefold purpose: an audit for the selected student, a repetition for all others, and a feedback for the teacher who is called "auditor" in this scenario. Before switching to an audit, the teacher is supposed to identify (and make electronically accessible to the selected student) the part in the slides which make up the subject to be (re)taught during the audit. As an auditor, s/he is supposed to take notes about the student's performance and about possible improvements to his own lecture. Figure 4 depicts the two situations, the document flow from slides to the audit subject, and one of the role flows: teacher → auditor (other role flows are omitted for brevity). Like in figure 3, the corresponding part of the scenario flow level is shown too; again, the information in brackets indicates – this time twice for two possible directions – the valid initiator for a mode switch (teacher), the fact that the transition is unconditional, and the names of the operations to be carried out, sw_audit and sw_class (these are at the same time the "events" which cause the transition); detailed information about the data flow is omitted.

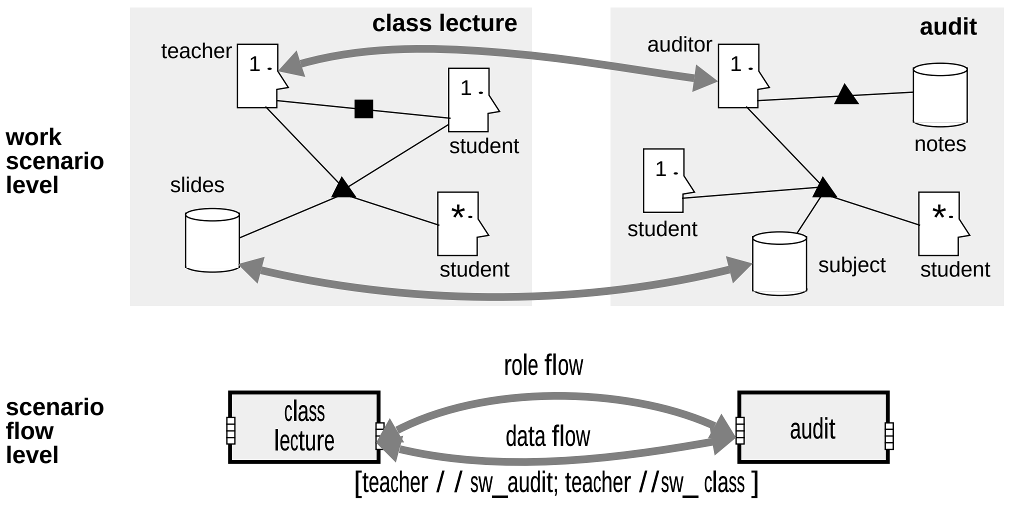

FIGURE 4. Scenario flow with changing situations

Note that role flows, in contrast to the “identical” link, indicate that the corresponding “user” nodes really change (switch) their role (even if the same identifiers are used as in the case of “student”). Note also that it is mainly the fact that teachers may switch back and forth between class lectures and audits which led to the decision to identify the CDUs “class lecture” and “audit” as different *situations* as opposed to different workflow type *steps*.

For an example of two different *steps*, we can return back to figure 2, i.e. to the example of lecture and groupwork scenarios. Figure 5 shows a primitive example with two lectures followed by a groupwork scenario. Instead of an event-based switching back and forth, a clear separation between the scenarios is assumed. In this case, one step is even supposed to terminate entirely before the subsequent one starts, so that links of type control flow are used.

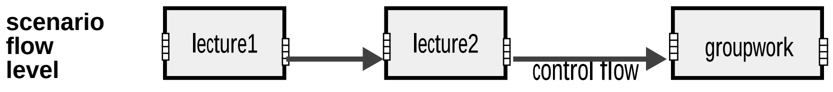

FIGURE 5. Scenario flow “teleteaching” with three steps

For all of the above examples, please note that a complete scenario flow will usually include a large number of CDUs, grouped at different levels where each grouping is of type fragment, situation, or step. Note also that many details have been left out in the examples above for brevity (further link types and links, more details about the links such as initiators, rights, flow elements and conditions, etc). According to the philosophy of the Items modeling technique, such details can be filled in at will without leaving “incomplete” designs inconsistent.

6 Status and Outlook

We have presented a high-level introduction to a design framework for complex cooperative applications which exploit advanced features like enterprise-integration, workgroup and workflow type cooperation, multimedia and multimodal interaction, and support for mobility and ubiquity. Consistently interconnected, scenario-based graphical views are used to keep the design of such very complex applications simple and understandable. The paper presented work in progress; a graphical multi-view editor is currently under construction. Large sample designs are used for evaluation.

References

[1] Blakowski, G., Hübel, J., Langrehr, U., Mühlhäuser, M.: *Tool Support for the Synchronization and Presentation of Distributed Multimedia,* Butterworth Jl. on Comp. Comm. 15 (10), 1992, pp. 611–618.

[2] Card, S.K., Mackinlay, J.D., Robertson, G.G. The Design Space of Input Devices. Blattner, M., Dannenberg, R. (Ed.): *Multimedia Interface Design*, Addison-Wesley, 1992.

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

[4] Green, J., Corman, P.: *Interactive Multimedia Association Architecture Reference Model.* Interactive Multimedia Association, Nov. 1992.

[5] Harel, D. On Visual Formalisms. *CACM*, 31(5), pp. 514–530, May 1988.

[6] Huston L., Honeyman P.: *Disconnected Operation for AFS*, Usenix Symposium on Mobile and Location-Independent Computing, Aug. 1993, pp. 1–10

[7] Johnson, J. Selectors: Going Beyond User-Interface Widgets. Proc. of CHI ‘92, Monterey, CA, Mai 1992, pp. 273–279.

[8] Kistler J. J., Satyanarayanan M.: *Disconnected Operation in the Coda File System*, ACM Trans. Comp. Sys. 10(1), 1992, pp. 3–25

[9] Multimedia in the Workplace. Special Issue, CACM 36 (1), 1993.

[10] Popek G. J., et al.: *Replication in Ficus Distributed File System*, IEEE Workshop on the Management of Replicated Data, Nov.1990, pp. 5–9.

[11] Pickering, J., Grinter, R. Software Engineering and CSCW: A Common Research Ground. In Taylor, R. (Ed.): *SW Engineering and Human Computer Interaction*., Springer LNCS, 1994.

[12] Roudaud, B., Lavigne, V., Lagneau, O., Minor, E. SCENARIOO: A New generation UIMS. Proc. of INTERACT ‘90, Aug. 1990, Elsevier Science Publ., pp. 607–612.

[13] Taylor, R., Johnson, G. Separation of Concerns in the Chiron-1 User Interface Development and management System. proc. of INTERCHI ‘93, Amsterdam, April 1993, pp. 367–377.

[14] Roseman, M., Greenberg, S.: GroupKit: A Groupware Toolkit for Building Real-Time Conf. Appl. Proc. CSCW ‘92.

[15] Rüdebusch, T.: *Development and Runtime Support for Collaborative Applications*, in: Bullinger, H.-J. (Ed.): Human Aspects in Computing, Elsevier Amsterdam etc., 1991, pp. 1128–1132.

[16] Steinmetz, R., Meyer, T.: *Modelling Distributed Multimedia Applications*. Int. WS Adv. Comm. and Appl. for High Speed Networks, March 1992, Munich.

[17] Schäl, T., Zeller, B.: Supporting Cooperative Processes with Workflow Management Technology. Tutorial on ECSCW ‘93, Milano, Italy, Sept. 1993

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