Providing for Organizational Memory in Computer-Supported Meetings

Gerhard Schwabe

Hohenheim University

Abstract

Meeting memory features are poorly integrated into current group support systems (GSS) . This paper discusses how to introduce meeting memory functionality into a GSS. The author first introduces the benefits of a good meeting and organizational memory to an organization. Then, challenges such as storing semantically rich output, building up the meeting memory during the meeting with a minimum of additional effort and integrating meeting memory into organizational memory, and privacy protection are discussed. Finally, using the Group-Object object-oriented model of a GSS, the author shows how meeting memory functionality can be implemented in a GSS.

1. Introduction

Organizational memory is the "stored information from an organization's history that can be brought to bear on present decisions" [29] or more detailed „we consider organizational memory to be the means by which knowledge from the past is brought ot bear on present activities, thus resulting in higher or lower levels of organizational effectiveness [33]. An organization needs information from its immediate past to support ongoing projects and coordinate its actions. It needs information from its history to provide models for future actions, to explain past decisions, to allow forecasts, to counter damaging myths, to function as a basis of change and to avoid repeating mistakes [25].

A significant amount of information relevant for future decisions or actions is generated in meetings. However, in many conventional meetings, the minutes and the memory of the individuals are the only records an organization keeps. As humans are poor at remembering the past [14] and particularly poor at remembering the rationale of their decisions [29], they are unreliable containers for a meeting memory. While it may not be feasible to persuade an organization to transfer its memory from individuals to a computer, the organization should make best use of the information captured by the computer to enhance organizational memory. Computer supported meetings offer an opportunity to store organizational memory in the computer, because in computer supported meetings much information is entered in the computer anyhow. Group support systems (GSS) introduce unique opportunities and challenges for keeping more extensive and flexible rec??ords of meetings. Yet, current GSSs such as GroupSystems [7], [17], [11] or VisionQuest [28] are ill prepared to store meeting information for a longer time. Morrison and colleagues [19], [20] tackle this problem by providing a database that can import and store meeting output. This paper goes one step further and shows how one can enhance the GSS itself with meeting memory features. To this end, the author has developed an object-oriented model of a GSS that collects data during a meeting in such a way that it allows easy access to all organizational context and content information.

The following sections discuss the unique opportunities GSSs provide for keeping better records of meetings. Challenges to record-keeping that are not completely met by current systems will also be addressed. Finally an object-oriented model of a GSS shows how these challenges can be met in a system design. A section on implementation, limitations and fututre research and a summary will close the paper.

2. Opportunities for improving organizational memory through GSS

Traditional meeting minutes are based on a meeting style that heavily depends on oral discussions. In some instances participants hand out documents beforehand or give a prepared presentation, but the heart of face-to-face meeting activities is oral. In order to collect a more elaborate meeting memory, one first has to change the meeting style to one that relies on "joint work on common material" [24], also called "common artifacts" [22]. Professional facilitators therefore introduce flipcharts, whiteboards, cards and other material [2], making meetings less talk and more work. These materials are distributed all over the room (in the hands of participants, on the walls, etc.) and participants work on them, sometimes alone (for example writing a card), sometimes in subgroups (on separate flipcharts) and sometimes as a complete group. In the course of the meeting, the entire room resembles the group memory. Anyone can refer to anything that he[1] or someone in the room has put on paper. Unfortunately, this group memory is destroyed once the group has finished its work and the room is prepared for the next meeting. Flipcharts with cards or drawings on them are difficult to archive, difficult to reuse, and impossible to share among the group members outside meetings. Photographs of them are a poor substitute with regard to their expressive power. Further, the whole work looses its context once it is removed from the room.

GSSs expand the idea of joint work on common material. The material can be shared in a more flexible way, participants can contribute in a true parallel fashion and in new workforms (e.g., anonymously or asynchronously) [24], [15]. This paper focuses on the memory abilities of a GSS: it automatically stores all information that has been entered in the course of the meeting and makes it accessible to anybody interested during and after the meeting. Once the information is both created and stored on a computer it is even possible to maintain much of its original context. Thus the group memory is not automatically destroyed after the meeting as in the flipchart-type meeting. It can be integrated into a project memory or an organizational memory, retrieved, reused and built upon by individuals, by the same group or by other groups of the organization. As the stored information has a context, anybody can reconstruct the history of the meeting-products creation and the rationale of decisions made in the course of a meeting. Thus a meeting memory does not only archive *what* has been produced, but also *how* it has been produced, *when* a contribution has been made, *who* has contributed and *why* decisions have been made[2].

Specifically a meeting memory can contain:

• *What* has been produced: the products of a meeting are archived in the form of texts, collections of comments, outlines, tables, drawings, votes, and diagrams. The products of the meeting contain what has been worked on, what has been agreed upon or what has been decided.

[1] The words „he“, „his“, „him“ are used solely for simplicity and better readability. They are not intended to stand for male dominance.

[2][29] also used the journalists questions how, why...

• *How* each product has evolved: the meeting memory collects information on the problem solving procedure used (e.g., brainstorming) for each agenda item, the activities (e.g., presentation, group writing, voting, outlining) and the content contexts (e.g., as an answer to a question, as a rebuttal, as a free-standing idea) of contributions. A meeting memory can even contain a history of all changes that have been applied to materials, if it stores multiple versions of products.

• *When* any product has been created: both the absolute time (e.g., June 1, 1993 1:25 PM) of any product's or contribution's creation, and the creation time in relation to any other contribution's creation time (i.e., has it been written before, during or after another contribution). Based on extensive timestamping information, a whole meeting can be replayed on the computer, making it much easier for new members to enter an ongoing activity. The duration of any activity indicates how much effort has been put into the activity. For example, a decision that has been made after a long discussion is probably much better grounded than a decision that has not been discussed at all.

• *Where* a meeting has taken place. Times and places trigger memories of the meeting content [21] and can serve as filters when searching for information.

• *Who* has contributed: archiving information about who has made a contribution gives the participants a sense of ownership and responsibility for meeting output. Contributions can be made by a single identified participant, an anonymous participant, the facilitator (on behalf of the group), several participants, a subgroup or the whole group. Authorship information is also important to judge the value of a contribution (e.g., from an expert). Last but not least, ownership allows each participant to take his personal cut at one or several meetings, filtering what people that interest him have contributed.

• *Why* a decision has been made: the more time that has passed since a decision has been made, the harder it is for an organization to find out why a decision was made the way it was, especially if the persons who originally made the decision have long since left the company. Without knowing the rationale of a decision, nobody can determine if a decision was a particularly clever one or if it was based on assumptions that are no longer valid. Even worse, without knowing the rationale of a decision one often is not able to follow the intentions of decisions even if one wants to. It is exactly the qualitative information needed to understand the rationale of decisions that is being exchanged in meetings. A meeting memory can make decision rationale explicit if participants have used special tools such as IBIS [16], [6]. It can also allow persons to reconstruct the rationale from the meeting output and the meeting context information [23]. The richer the meeting documentation the easier it is to reconstruct the rationale of decisions.

3. Challenges to GSS

3.1. Storing rich output

Why not just store the GSS-output as documents and forget about the rest? One basic argument has been elaborated on in the previous paragraphs: reducing GSS-output to conventional documents takes the contributions out of their context. Looking at typical meeting minutes, one senses how much effort the writer goes through in order to reconstruct the meeting context (describing who said what when how).

The other argument against storing GSS output as conventional documents is that typical meeting output does not resemble a conventional document, but looks more like a hyperdocument structure [13]. Take for example a meeting in which a group first brainstorms and discusses ideas with a brainstorming tool, then organizes the ideas with an outliner, and finally votes on selected issues. The gathered information resembles a network in which several structures overlay one another: brainstorming typically organizes information into several sheets of comments. Comments on these sheets can refer to one another, i.e., have content links. Some of these comments are linked to an author, others are anonymous. Afterwards, the comments are organized into the treelike-structure of an outline. It would be useful not to be forced to destroy the structure of the brainstorming output by this reorganization and at the same time avoid having to duplicate the information. Duplication of information (e.g., copy all brainstorming remarks before reorganizing them into an outline) tends to make the information inconsistent and makes search for a particular comment much more burdensome (i.e., which copy do you want?). If an issue has been voted on and this issue was an ele ment of an outline, a user should be able to access the voting information from the outline.

Thus the meeting memory has to be stored in finer grains than conventional documents. According to the experiences of the author, a comment or paragraph as an atomic unit for meeting output that is mainly textual is a fair compromise between the needed flexibility and the danger of "getting lost in hyperspace". A paragraph is the smallest unit of text that makes sense in itself. Using words or sentences as atomic units leads to a large overhead for storing information and threatens the performance of a system. Using "speechacts" [23] places an excessive additional burden on the participants, because it requires participants to make their intentions explicit all the time. These basic units need to be linked to other content information and to context information such as authors, dates and agenda items.

3.2. Building up memory with a minimum of additional effort

Participants want to concentrate on the meeting itself. They have little understanding for additional work that is not directly related to the meeting purpose. Asking participants to organize information in a particular way required by an organizational memory system (e.g., into flat tables of a relational database) is not acceptable. The challenge therefore is to make best use of the data that is collected in the course of the meeting anyway. For example, timestamping information can be collected automatically in the background and authoring information requires one login of each participant at the beginning of the meeting. A detailed discussion of information that can be automatically collected in an GSS will be given in the discussion of the object-oriented model of GSS.

3.3. Making information easily accessible and maintainable

Organizations that memorize their history must also be able to organize, access and forget this memory in large chunks (organizations may be forced to forget information e.g., by laws protecting privacy). While it is necessary to store very fine-grained information, it must be possible to find and recover large chunks of information (e.g., all meeting output of a project) for further inspection or for deletion. As meeting work and meeting decisions may directly influence day-to-day work, information should be accessible from the desktop.

Generic techniques for finding information in unstructured or semistructered documents are discussed in information retrieval. Keyword indexes or search trees impose an artificial order within and across documents; query languages allow sophisticated questions. These techniques can be applied to meeting memories and are useful for retrieving meeting information from an organizational memory [4]. Including meeting information into a database that allows flexible search mechanisms becomes paramount, once a larger institution archives meeting information for any length of time. While this requirement might look obvious, typical Group Support

Systems (e.g., GroupSystems, VisionQuest, SAMM) are neither built on top of nor provide an interface[3] to database management systems. The main difficulty lies in the structure of traditional relational database management systems that do not support storing data of arbitrary length and type as is typical for meeting output. Extended relational DBMS [26], object-oriented DBMS [1] and office form DBMS (e.g., Notes [9]) should improve possibilities to archive meeting data in DBMS and to integrate meeting information with other office information.

3.4. Integration with other organizational memory data

Meeting output needs to be integrated with other office information. A prerequisite for integration is a network connection between the meeting environment and the office environment. The easiest way to integrate meeting output into the personal office environment is to convert the output to documents and allow access to the documents via a central file server. This approach has several disadvantages: converting meeting output to documents takes the information out of its context. Participants might loose valuable clues for understanding the meaning of a contribution and it is much harder to synchronously or asynchronously continue a meeting. If one duplicates the output keeping a snapshot of a meeting for possible continuation in the GSS and a conventional document in the office environment as organizational memory, both copies quickly get out of sync. Furthermore, file systems have only rudimentary access and search mechanisms. Another approach is keep only the GSS meeting snapshot and to access the GSS directly from the individual office. This allows "anytime-anyplace" access to the meeting memories, but it does not integrate the meeting memory with the rest of the individual data and the organizational memory. A lack of integration leads to duplication of data, inconsistency of data, longer search times, increased teaching efforts and loss of links between meeting data and other data.

There are three mechanisms that allow for a closer coupling of a given meeting memory and other information:

External links connect meeting contributions with office information via a mechanism of the office information system. For example Microsoft Windows supports information linkages with OLE (Object Linking and Embedding). In theory a piece of information (e.g., a comment) embedded in another object (e.g., a document) is updated, the embedding object reflects these changes immediately. Current external linking mechanisms have only rudimentary synchronization facilities and basically target single-user-support. As the navigational features of external links (how does one find information) and its security features (how do I prevent somebody else from (unintentionally) removing objects that I need) are poor, external links remain problematic as a basis for an organizational memory.

Internal links connect meeting outputs with office information via a mechanism of the GSS. Any information that is related to a meeting is imported into the GSS and becomes an integral part of the meeting memory[4]. While internal links work well during the meeting itself, they are very cumbersome for an organizational memory. Very soon, a lot of information that is produced in an organization is being used both in and outside meetings. During a meeting one can only use the tools of a GSS; outside the meeting one wants to use personal tools (such as word-processor). Typically this conflict results in a vast duplication of information that again almost certainly will be inconsistent. If one keeps only a reference to related office information in the meeting memory (e.g., storing the file-name), the same problems as with external links apply.

Integrated organizational memory seamlessly embeds the meeting environment into the office environment and the meeting memory into the organizational memory. The basis for an integration is a DBMS that contains and organizes both meeting data and other office information. This DBMS imposes a common organizational structure upon both meeting and individual data. Typically an organization collects data under projects or departments. Each member can then access and work on all of his information in meetings and in his normal office environment. If a person wants to seamlessly move from individual work to groupwork and vice versa, the DBMS must store a common representation for output produced by personal tools and group tools. Optimally the same set of tools works for both individual and group work. GO (described below) follows the integrated organizational memory model, although it is probably unrealistic to expect an organization to switch to the needed new DBMS. However, the author believes it is better to start with a clean solution and later make compromises in the actual organizational implementation than to design for compromise from the very beginning.

[3] except for ASCII export

[4] This is the approach of current GSS, e.g., GroupSystems

3.5. Protecting privacy

Archiving who said what, when, why, and to whom offers good opportunities for a better organizational memory . In many instances the stored information is intended to be freely shared by any interested persons within the organization. However, in other instances meeting information should remain confidential to the group, shared with only part of the group or purposefully only mentioned orally and not put on paper. Conventional meeting minutes are carefully crafted documents containing only what they have been agreed to contain (or what the minute-taker thinks the group has agreed upon). A GSS that interferes with these habits and acts like a "Big Brother" will not be accepted. On the other hand, allowing for anonymous contributions turned out one of the most important features of GSS in the eyes of the participants [8], [27]. Its additional privacy protection can be one of the major facilitators for introducing GSS to business groups. Thus a meeting memory should not only protect sensitive information and the privacy of the meeting participants as it is already being done in a conventional meeting. It should also make best use of information technology to enhance confidentiality where it is necessary to allow for a free discussion or to protect the interests of the individual. Measures can be taken before, during and after a meeting.

● Pre-planning for privacy before the meeting: privacy issues should be contained in the pre-planning of a meeting. A facilitator plans for each agenda item whether it is discussed anonymously, semianonymously or openly. At the beginning of the meeting the facilitator briefly introduces new participants to the meeting memory and privacy-protection capabilities of the system.

Privacy measures during the meeting:

● Allowing pseudonyms: any person who wants to keep his participation confidential, logs in under a pseudonym. Contributions by a pseudonym person are owned by the group.

● Duplicate control of anonymity: during the meeting anonymity, awareness (who is doing what now) [10] and authorship (who has contributed what in the past) remains under the control of both the facilitator and each individual member. Leaving anonymity under facilitator control is cumbersome, because participants might want to quickly switch between different anonymity modes. Leaving anonymity under participant control only can endanger anonymity, because a single individual cannot remain anonymous on his own, if all other participants contribute openly. There are also instances where a facilitator wants to enforce anonymity in order to avoid group pressure for openness (as it happened in the east block, where all citizens "voluntarily" voted open for the communist party). Leaving anonymity under duplicate control means that a contribution remains anonymous if one (i.e., the facilitator or the participant) wants it this way.

● Leveled precision of authorship information: for protecting privacy in the meeting memory the control of authorship is important. To reconcile the need to understand information out of its context and the need to protect privacy a leveled approach to authorship is useful: the (real) author can decide if he wants the contribution to be stored under his individual authorship (e.g., author: John Miller), his subgroup authorship (e.g., author: "the soap marketing task force"), his group's authorship (e.g., author: "the marketing department" that has gathered for a meeting at this moment), or his organization's authorship (e.g., author: "the Perfume Company"), or absolute anonymity (author: "Anonymous"). Usually each participant does not decide each contribution's authorship on its own, but he uses default values that he only changes in a few instances.

● Strong and weak memory: in a typical meeting there are contributions that are explicitly made for the records and other contributions that are not intended to leave the room. There should not be any records of such information, because its author does not want to be committed to it. A meeting memory can support "contributions without trace" by differentiating between weak memory and strong memory. Strong memory contains all information that is to be archived; information in weak memory is automatically deleted once the meeting is over. The decision if a contribution belongs to weak or strong memory can be made at any time in the meeting, e.g., when the contribution is made, after the related agenda item has been discussed, after it has been voted on, or at the end of the meeting. Strong and weak memory may not protect participants within a meeting from one another, as any participant may very easily make a copy of the weak memory, but it protects the group members from persons outside the meeting. Thus a spirit of confidentiality is supported.

Privacy measures after the meeting:

● Access rights: just as any other office information the meeting memories in the organizational memory must be protected against misuse and sabotage. Traditionally DBMS and operating systems take care of data protection by giving access privileges to selected persons only. While this requirement seems obvious the current situation typically is different as GSS are not connected to DBMS (see above).

● Formal meeting minutes: in conventional meetings often one person has the role of a minute-taker. During the meeting he takes notes of the participants' contributions. After the meeting he goes through his notes and tries to find out what the group has done and agreed upon. In this process he filters all comments that he thinks participants want to be "off the records". Afterward the meeting minutes are distributed to the participants and the minutes are formally agreed upon. Only then do the minutes become part of the organizational memory. A similar post-meeting review of the meeting output can be done in computer supported meetings: a minute-taker is chosen and goes through all the contributions after the meeting correcting or deleting problematic or irrelevant comments. This condensing process can even result in the usual brief meeting minutes[5]. He then distributes the meeting output to the participants allowing each participant to voice objections. Only then does the meeting memory become part of the organizational memory.

● Fuzzy timestamps: timestamping information can corrupt other privacy measures as other information can be reconstructed from timing information. Storing the exact time of the contribution makes it possible to pin down the meeting and the persons who participated in the meeting. While detailed timestamping information can be important during the meeting for synchronization of work, the storing of detailed timestamping information can be relaxed after the meeting. One can store fuzzy timestamps. Storing only the hours or date of a contribution protects the individual participant within the meeting from being identified with a comment. Storing only relative timestamps (starting a clock at 0:00:0) helps to protect a contribution from being identified with a certain meeting. Relative timestamps keep enough information to replay a meeting.

● Control authorship: authorship of contributions does not necessarily need to stay the same after a meeting as it has been during a meeting. A GSS can allow participants to generalize authorship after the meeting if they do not want to be identified with a contribution outside the group. A minute-taker or any other participant can generalize authorship by making individual contributions into subgroup contributions, subgroup contributions into group contributions and group contributions into organizational contributions or making a contribution anonymous all together.

[5]even if this would destroy many of the enhanced organizational memory features.

How can privacy protection be reconciled with building up organizational memory with a minimum of additional effort? Do not the strategies of pseudonyms, duplicate control of anonymity, controlled authorship, strong and weak memory, and fuzzy timestamps put too much of an additional cognitive burden on the participants? One can assume that an additional cognitive burden is acceptable to the participants in two cases: (1) if they must perform the actions only once in a meeting or (2) if the actions are to their own interest and do not have to be performed continuously. Participants have to log in with a pseudonym only once in a meeting, fuzzy timestamps are applied to all meeting output only once after the meeting and strong and weak memory has to be cleaned up only once after the meeting. Individual control of anonymity, control of authorship, and marking weak and strong memory are in the interests of each participant. Working with default values that are preset by the facilitator for a meeting or session and which can be easily overridden by each participant should be a feasible approach.

4. An object-oriented model of GSS

The author has developed the GO ('Group Objects') GSS-architecture and a GO-prototype that take organizational memory requirements for GSS into account [25] and shows how a GSS designer can integrate meeting memory features into a GSS. The GO-architecture is an object-oriented model of a GSS. GO is based on numerous observations of actual and experimental computer supported sessions[6] and an analysis of current GSS. The GO-prototype is a Smalltalk-80 program intended to be used in conjunction with an object-oriented database management system. The following paragraphs very briefly describe the object-oriented description language and then discuss those parts of the GO-architecture that are relevant for meeting memory and organizational memory.

4.1. The object-oriented description language

GO is described using the Coad/Yourdan notation [5].

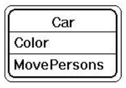


Fig. 1: Class&Objects

Class&objects stand for existing or conceptual unities, e.g., 'Tom', 'Tom's VW', 'The United States' or 'Anti-Apartheid legislation'. To be more precise, class&objects serve as "cookie cutters" for such unities, also called *instances*: a person class&object serves as a cookie cutter for the *instance*

[6]conducted at the University of Arizona, the University of Michigan and Hohenheim University.

'Tom', a car class&object serves as cookie cutter for the instance 'Tom's VW' and so on. Class&objects[7] are described by their *attributes* which describe their features and their state and by their *services*, which make known what the object can do for you. For example a car has the attribute color and a car offers the service 'move persons'.

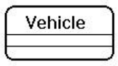


Fig. 2: Classes

Classes model conceptual unities like class&objects with the only difference that they are purely abstract. They are cookie cutters but there are not supposed to be any cookies produced with them. They only make sense in conjunction with gen-spec structures.

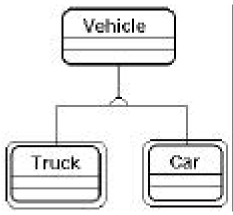


Fig. 3: Gen-Spec-Structures

Gen-spec structures describe an abstraction hierarchy between classes. The higher class is a *generalization* of the lower class, or, in other words, the lower class is a *spec*ialization of the higher class. A vehicle is a generalization of a car (because there are other vehicles such as trucks or carriages) and a car is a specialization of a vehicle.

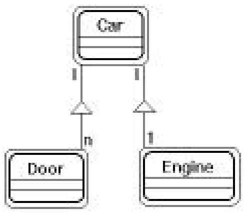


Fig. 4.: Whole-part structures

Whole-part structures describe the relationship between a whole instance and its parts. For example, the parts of a car are its engine, the doors, the seat belts, etc. It is normally not so easy to differentiate between whole-part structures and instance connections. A rule of thumb says that there exists a whole-part relationship if the parts cease to exist with the whole, e.g., if your car burns, the engine is destroyed, too.

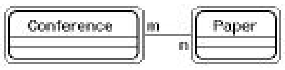


Fig. 5: Instances connections

Instances connections model all structural relationships between class&objects. Just like in ERM-Models, there exists (1:1, 1:n and m:n) relationships. For example, there is an m:n relationship between conferences and authors as authors hand in papers to several (n) conferences and a conference receives several (m) contributions by authors.

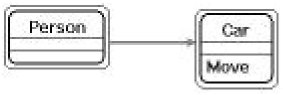


Fig. 6: Message connections from a car.

Message connections stand for service requests by one class&object to another class&object, e.g., a person can request the service 'move'

[7]There is no difference to the term 'class', normally used in object orientation.

4.2. The GO-Architecture and how it helps designing organizational memory

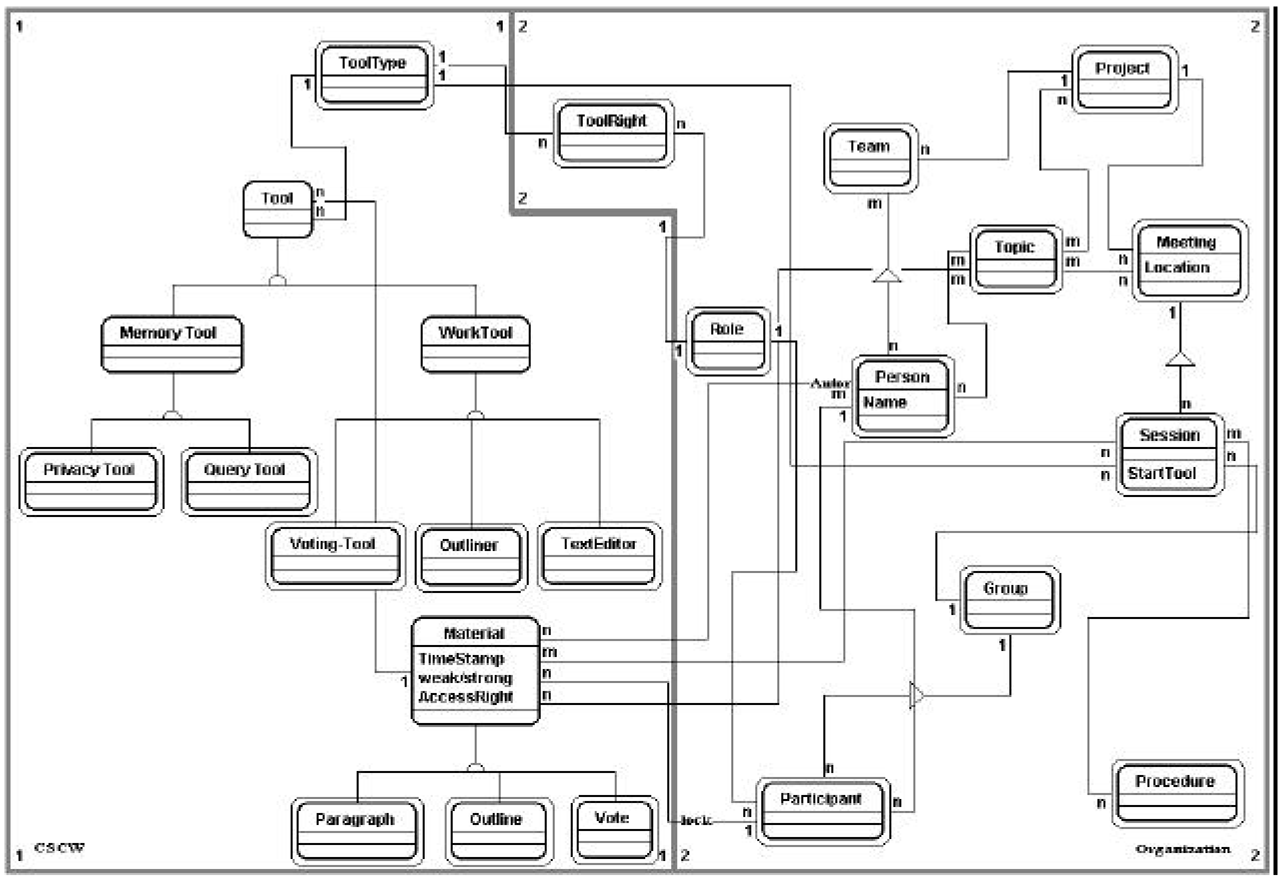


Fig. 7: Overview over basic structural components of the GO architecture)

The model consists of two parts: the left part (*subject1*) is called the 'CSCW-part' as it describes sharing of information among people ; the right 'organizational part' (*subject2*) consists of objects that are typically needed to run a meeting in an organizational setting. The attributes and services of class&objects just list a few features and activities necessary for the understanding of the model and are by no means complete.

4.2.1 CSCW part

The CSCW-part centers around the class&objects *tool*[8] and *material* [3] (fig.8). Participants use tools to manipulate their material. Tools can be for instance *work tools* such as *outliners*, *voting tools* or simple *text editors* or *memory tools* such as *privacy tools* or *query tools*. Material can be a simple text, an *outline*, an outline-node, a *paragraph*, a graph, a drawing, a matrix, a matrix-element, a *vote* ..... The subclasses of tool and material serve as examples for a much larger potential range of text based and graphic based tools. Note that neither tools are the same as (PASCAL-) procedures nor material is the same as data: tools have their own state e.g., a text editor knows where to insert the next character, and their own attributes, e.g., the preferred font. Material is not plain data as it has the sole knowledge of how it should be manipulated, e.g., a tree knows how to insert a node or any material knows

[8]All class&object names in the following paragraphs are marked italic, when they are introduced for the first time.

how to broadcast the fact that it is locked.

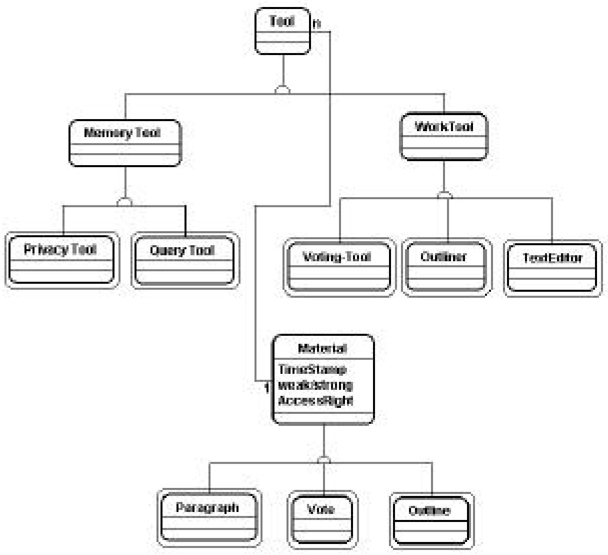


Fig. 8: Tools and material

The dependency between tools and material is one-way: a tool knows the interface-procedures of a material; the material does not have any information about the tools that work on them [12]. To make this idea plausible: the meat does not need to know anything about the knife in order to be cut, but the knife needs to take the features of the meat into consideration in order to be able to cut it. Groupwork is then seen as common work with individual tools on a shared material. This way one is able to build all kinds of different tools without ever having to change the material.

A major benefit of a clean separation between tools and material lies in the fact that the meeting memory only needs to store the material and forget the tool information as soon as the tool is not needed any more. 'Material' contains the information, you want to store and administrate in an (object-oriented) DBMS[9]. 'Tool' contains the information that should be stored locally, if at all. Storing material in a DBMS provides the whole DBMS support in locking, synchronization, distribution, security, and query functionality fulfilling many of the requirements mentioned above.

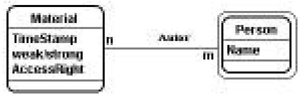
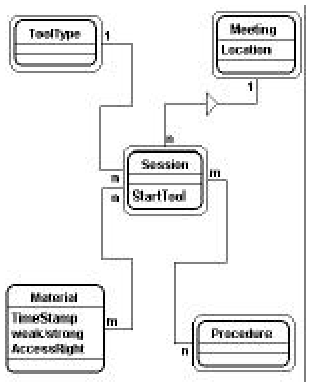


Fig. 9: Person and material

Creating one abstract class 'material' with concrete subclasses (outline, vote...) concentrates all the authoring, locking, timestamping and weak/strong memory administration in one place: the material (fig. 9). Explicitly storing authoring information of a node protects privacy with access control and to search for what a given person has contributed (for a deeper discussion see [32]. Timestamps allow for searching when and in what order contributions have been made; fuzzy timestamp mechanisms can be implemented at one place. The privacy protection scheme of strong and weak memory can be reduced to one simple attribute ('weak/strong') and a garbage-collector that is automatically started at the end of the meeting. The privacy tool allows participants to control privacy during and after the meeting. Since material can be as large as a document with thousands of pages and as fine-grained as one paragraph[10] or one drawing element, it is rich and easily accessible and maintainable.

Organizational part

The organizational part (right side) supports administration of meetings and puts group work into an organizational context. As argued above, the meeting context is an important part of the organizational memory. Specifically one wants to store the agenda, i.e., the *sessions*, a *meeting* consists of (fig. 10). Each agenda item is linked with the material that has been produced during this session. If the session has been part of a problem solving *procedure*, this information is stored, too. The link between a session and a *tooltype* remembers how the material has been created.



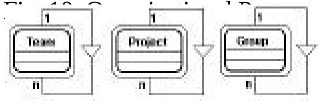


Fig. 11: Projects

A hierarchical structure of *projects*[11] collects meetings into ever larger organizational units (fig. 11); projects also integrate material that has been produced outside a meeting, if a person prefers to use his personal tools. The meeting context gives clues how the product has evolved; storing the agenda, linking the sessions with their material and procedures helps to find out later why a decision has been made.

[9]Object Oriented DBMS store class&objects with all their attributes and services, i.e., they are able to store the complete material, not only the data part.

[10]Or even more finegrained; the administration overhead is huge though if one, for instance, uses words as finest grain.

[11] The hierarchical structures of project, group and team are left out in all other drawings to increase readibility.

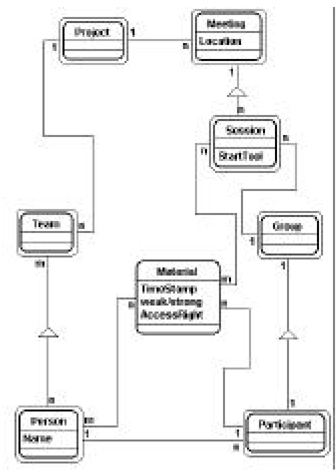


Fig. 12: Groups and teams

GO strictly separates between standing groups (called *teams*) and acting groups (called *groups*) [18] (fig.12). Teams and their members (called *persons*) represent the long-term organizational structure; groups and their members (called *participants*) represent the actors present in the meeting room. Teams and groups frequently overlap but are not necessarily identical. In a meeting, a member of the team can be missing or an outsider can be invited to join the group. Information on groups is needed to run the actual meeting, e.g., for session management or for awareness of what others are doing. Information on teams is interesting for organizational memory, e.g., who has contributed, what project does the work belong to. While persons are identified by their name, participants need only be internally identified by their workstation number.

During the meeting, participant information (e.g., participant1) and person information (e.g., Tom) is linked. The link allows participants to be aware where people work; e.g., Tom can see in his outliner that Kathy is working on the last paragraph of the shared outline (the system follows the link from participant data to the person data to provide the information). The person information is logged by the GSS; the participant information leaves no trace. While Kathy is working on a paragraph of a shared outline, both her participant object and her person object is connected to the paragraph material. After she has finished, the link to her participant object is cut, but the link to her person object (called authorship) is kept for organizational memory. After the meeting the link between participant and person objects is cut and there is no other way to trace a contribution to its author than by the stored person information. The separation between person and participant information allows for a clean separation between session management and organizational memory management. The stored person object (e.g., Tom) can freely be manipulated during and after the meeting without affecting the system performance in the meeting: whenever a group member wants to generalize authorship, e.g., make a contribution group-owned or totally anonymous, he just relinks his participant to another pseudo-person. This pseudo-person can have the name of a subgroup, group, organization or 'Anonymous'. Since the linking information is not stored, there is no way to trace the actual identity of an author of a comment after he has relinked to his original identity. Hiding participant information also protects the privacy of all group members who have logged in under a pseudonym.

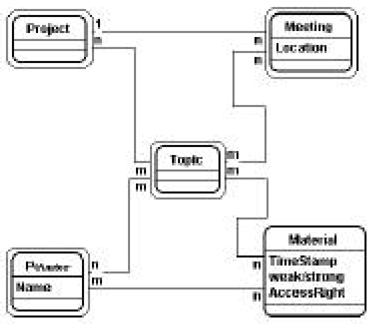


Fig. 13: Topics

There remains one problem to be addressed: an organizational memory that has been growing over the years should serve as an encyclopedia. Encyclopedias organize information by topics that cut across time, space and persons. Thus there needs to be a *topic*-class&object that serves as a container for all meetings, projects and materials regarding a particular topic (fig. 13) [19], [20]. In GO, topics also contain a reference to persons who have identified themselves as expert on a particular topic. One can use this person information to scan through meetings in which the expert has participated.

5. Implementation, Limitations and Future Research

The GO-architecture has been successfully implemented in Objectworks Smalltalk prototype (described in detail in [25]). This prototype is running on a single computer simulating several participants. However, the prototype has never been tested in a real meeting. Currently we are leading the research project CUPARLA supporting the collaboration of the Stuttgart City Council [30][12]. In this project meetings are supported with GroupSystems and asynchronous collaboration is supported with Lotus Notes. GroupSystems has an interface that stores meeting output in a Lotus Notes database. The participants can continue to work on the meeting documents in Lotus Notes and reload the documents into GroupSystems. Collaboration in a city council raises the organizational memory and privacy issues discussed so far. We therefore currently work on including organizational memory features into Lotus Notes databases and applications. Using Lotus Notes comes close to the described ideal of an integrated organizational memory. However some new issues have appeared during the implementation efforts:

[12] Cuparla is part of the R&D Programm of the DeTeBerkom GmbH, a 100% subsidiary of the German Telekom.

(1) Collaboration support has to be very easy to use and to make best use of the learned working behavior. This behavior is very context dependent. For example, there is much more openness in meetings within the same party than in meetings with members from different parties. We therefore provide software „rooms“ [31] as a working context, e.g., a party room. Rooms give access to information for some specific group and protect its privacy from other groups. The first usage experiences are encouraging.

(2) Once the organizational memory grows finding information becomes an issue. Particularly awareness of what has changed is important. We are experimenting with intelligent agents to identify changes to information that are relevant to particular users. An event mechanism will then notify the users of the changes.

6. Conclusions

This article has first described how collecting organizational memory information during meetings benefits an organization: it allows participants to access what has been done and agreed upon, when and how, and by whom. With this information, a participant can reconstruct much more of the meeting context than in conventional meeting minutes or the document output typical of current GSS. The meeting context will help to find out why decisions have been taken. Challenges that organizational memory pose to GSS include storing rich output, allowing participants to build up the meeting memory with a minimum of additional effort, making information easily accessible and maintainable, integration with other organizational memory and protecting privacy. Afterwards an extract of the GO object-oriented model has been explained that shows how a GSS designer can cope with challenges posed by organizational memory in a often surprisingly simple way. Organizational memory requires database functionality of GSS. The longer an organization will use a GSS, the more important this database functionality will become. Thus systems that store the meeting information in a DBMS, integrate meeting memory into organizational memory, make meeting information available to people outside meetings and allow them to coordinate their work via sharing material and context look like the GSS of the future to the author. First steps toward the realisation of this aim are currently taken in the CUPARLA -project.

6. Literature

[1] Atkinson et al. The object-oriented database system manifesto. *Proceedings of the Conference on Deductive and Object-oriented Databases (DOOD)*, Kyoto 1989, 40-57.

[2] Bostrom, R.; Anson, R.; Clawson, V. Group Facilitation and Group Support Systems; *workingpaper* Department of Management, University of Georgia, Athens (July 1991).

[3] Budde, R.; Züllighoven, H. *Softwarewerkzeuge in einer Programmierwerkstatt. Berichte der GMD.* München: Oldenbourg, 1990.

[4] Chen, H.; McHenry, W.; Lynch, K.; Goodman, S. Computer-Based Support for Organizational Memory: Framework and Design. *Workingpaper* 91-37, Department of MIS, University of Arizona, Tucson (1991).

[5] Coad, P.; Yourdan, E. *Object-Oriented Analysis.* Englewood Cliffs: Prentice Hall, 1991.

[6] Conklin, J.; Begeman, M. gIBIS: A Hypertext Tool for exploratory policy. *ACM Transactions on Office Information Systems*, 6, 4, (October 1988), 303-331.

[7] Dennis, A. et al. Information Technology to Support Electronic Meetings. *MIS Quarterly*,12,4(1988), 591-624.

[8] Dennis, A. *Parallelism, Anonymity, Structure and Group Size in Electronic Meetings.* Dissertation in the Graduate College of the University of Arizona, Tucson (1991).

[9] De Jean, S. Lotus Notes, Version 2.0: Connectivity and New Pricing Broaden Its Appeal. *PC Magazine*,10,(May 1991), 33-35.

[10] Dourish, P.; Belotti, V. Awareness and Coordination in Shared Workspaces. *Proceedings of CSCW '92 (October 31 to November 4).* Toronto 1992, 107-114.

[11] No Author: *GroupSystems Manual, Basic Tools Version 4.0,* Tucson:Ventana Corporation, 1990.

[12] Gryczan, G.; Züllighoven, H. Objektorientierte Systementwicklung - Leitbild und Entwicklungsdokumente. *Informatik Spektrum*,15,5(Oktober 1992),264-272.

[13] Hahn, U.; Jarke, M.; Eherer, S.; Kreplin, K. CoAUTHOR-A Hypermedia Group Authoring Environment. Bowers, J.; Benford, S. (ed.) *Studies in Computer Supported Cooperative Work*, North-Holland: Elsevier, 1991.

[14] Huber, G. A Theory of the Effects of Advanced Information Technologies on Organizational Design, Intelligence, and Decision Making. *Academy of Management Review*,15,1(1990),47-71.

[15] Krcmar, H.; Lewe, H.; Schwabe, G. Empirical CATeam Research in Meetings. Nunamaker, J.; Sprague, R. (ed.) *Proceedings of the 27th Hawaii International Conference on System Sciences*, Washington: Computer Society Press, 1992.

[16] Kunz, W.; Rittel, H. Issues as Elements of Information Systems, *workingpaper* Nr: S-78-2 des Institut für Grundlagen der Planung igp Stuttgart 1970.

[17] Lewe, H.; Krcmar, H. GroupSystems: Aufbau und Auswirkungen. *Information Management,* 7(1992), 32-41.

[18] McGrath, J. *Groups: Interaction and Performance*. Englewood Cliffs: Prentice Hall,1984.

[19] Morrison J.; Morrsion M.; Vogel, D. Software to Support Business teams. *Group Decision and Negotiation*, 2(1992), 91-115.

[20] Morrison J. Team Memory: Information Management For Business Teams. *Proceedings of the 26th Hawaii International Conference on Systems Sciences (Jan. 5-8, 1993)*, Hawaii: Wailea, 1993.

[21] Newman, W.; Eldridge, M.; Lamming, M. PEPYS: Generating Autobiographies by Automatic Tracking. *ECSCW91 (24-27 September 1991)*, Dordrecht, Boston, London: Kluwer Academic Publishers, 1991.

[22] Robinson, Mike. Keyracks and computers: an introduction to "common artefacts" in Computer Supported Cooperative Work. *Wirtschaftsinformatik*,35,2,(April 1993), 157-166.

[23] Sandoe, K.; Olfman, L.; Mandviwalla, M. Meeting in Time: Recording the Workgroup Conversations. *Proceedings of the 12th ICIS (December 16-18),* New York, 1991, 261-272.

[24] Schwabe, G. Computerunterstützte Sitzungen. *IM-Information Management*, 9,3(July 1994), 34-43.

[25] Schwabe, G.: *Objekte der Gruppenarbeit.*, Gabler, Wiesbaden 1995.

[26] Stonebraker, M.; Kemnitz, G. The POSTGRES Next-Generation Database Management System. *CACM,* 34, 10(August 1991), 78-92.

[27] Valacich, J.; Jessup, L.; Dennis, A.; Nunamaker, J. A Conceptual Framework of Anoymity in Group Support Systems. *Workingpaper* 91-27 of the Department of Management Information Systems, College of Business and Public Administration, University of Arizona, Tucson 1991.

[28] No Author: *Visionquest Users Guide;* Collaborative Technologies Corporation, Austin, Texas 1991.

[29] Walsh, J.P., Ungson G.R. Organizational Memory. *Academy of Management Review*,16,1 (January 1991),57-91.

[30] Schwabe, G.; Krcmar, H. Telearbeit im Stuttgarter Stadtparlament - erste Erfahrungen. *Proceedings Telearbeit 96*, empirica, Bonn 1996.

[31] Henderson, D.; Card, S.: Rooms - the Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-based Graphical User Interface. *ACM Transactions on Graphics,* Vol. 5, Nr. 3 Juli (1986), 211-243.

[32] Mandviwalla, M.; Sandoe K.; Clark, S. DII Collaborative Writing as a Process of Formalizing Group Memory. Nunamaker, J., Sprague, R. *Proceedings of the Twenty-Eighth Annual Hawaii International Conference on System Sciences*, IEEE Computer Society Press, Los Alamitos 1995, 342-351.

[33] Stein, E. , Zwass, V.: Actualizing Organizational Memory with Information Systems. *Information Systems Research*, 6, 1995, 85-117.