Design Versioning – Problems and possible solutions for the automatic management of distributed design processes

Sven Schneider, Jörg Braunes, Torsten Thurow, Reinhard König
Bauhaus-University Weimar, Germany

Designing is a complex process. Where this process involves multiple participants located in different places, digital tools for supporting this process are indispensable. However, the use and creation of tools for supporting design processes necessary entails intervening in or manipulating the process it intends to support. For design collaboration tools the coordination mechanisms employed are a crucial aspect. To make these mechanisms as flexible as possible, the technical challenge lies in devising an adequate concept for storing the actions that happen during designing. This paper deals with the issue of versioning in computer-supported collaborative design environments. The paper examines the technical and conceptual problems of versioning and discusses possible solutions.

Organizing Collaborative Design

Design processes are typically characterised by the division of labour. In numerous design steps, various aspects (formal, functional, environmental, economic, legislative, the urban realm, etc.) must be considered and coordinated. In addition, these aspects are perceived and evaluated differently by different stakeholders (architects, planners, future users, local residents, conservation authorities, developers, authorities, etc.) [1]. The central challenge in collaborative planning processes is to integrate the different aspects and interests at an early stage so that the skills and knowledge of the participants are used most effectively. For economic reasons, collaborative processes are no longer bound to a particular place

and time but increasingly involve participants in different locations, who do not necessarily work concurrently. The success of such distributed collaboration processes depends to a large degree on the media used (see for example [2]). The quality of the process affects the quality of the design outcome: for example, how do the different participants' contributions enter the process? Are their intentions recognized by other parties? How do these parties interact with each other? Effective communication and coordination between the different actors is essential in order to usefully support the collaborative process.

Because of the complexity of design processes, innovative management approaches are required for successful creative collaborative work [3]. If we consider the latest technical developments in CAAD systems, the concept of building-information modelling (BIM) does indeed focus on the integrative aspects of a common model. Nevertheless, it still adheres to old working patterns. BIM systems concentrate mainly on the effective exchange of data and necessitate a strict division of roles for distributed plan editing (see next section). This is most useful in the advanced stages of a project, because lessening the loss of information helps prevent misunderstandings, in turn minimising the incidence of extra costs further on down the line. However, when developing ideas at a conceptual level, where decisions can have a major impact on the later project, the pure exchange of data is of limited use. Although the exchange times between the different actors are shorter, the process itself is still strongly sequential. But in the early stages of designing one is typically dealing with so-called 'wicked problems' [4] "a priori" hierarchically organised structures can hinder the emergence of creative solutions and innovation.

Creative group work is typically a highly dynamic process where the participation of different actors does not follow a pre-defined pattern, the actions are often unpredictable and produce a wide variety of emerging solutions. In addition, there are no clear start and end conditions. The product and its design goals, general conditions and organisational structures often only surface during the design process. *"The major challenge to CSCW emerging from this is the very fact that ordering systems are constructed and maintained in a cooperative process"* [5]. For this, new management methods are required that are able to accommodate the particular characteristics of creative processes [3]. When developing these methods, it is important to keep design processes open, to motivate actors to interact with each other, and correspondingly to reduce technical barriers that could hinder the effective integration of the different actors' skills. To coordinate a network that encompasses different tools, contents and users, it is necessary to map the actions taking place within it. This requires the storage and versioning of the entire development process. To

adequately integrate such a versioning concept into the collaborative design process, different design-specific characteristics have to be taken into account:

(1) Designers communicate using representations of the object to be designed (sketches, models, drawings, text, etc.). These representations are interpreted, reviewed and continually improved [6].
(2) Different tools are used to edit these representations. Due to the limited functionality of each tool, designers switch tools depending on the respective need.
(3) Variants and alternatives emerge during the editing process. In addition, several variants and alternatives can exist in parallel at a point of time.
(4) The time at which a particular design state is reached cannot be clearly defined. Intermediate results can also trigger creative processes and discussions within a group (see, for example, "the reflective nature of designing" [7], and "thinking via perception" [8]).
(5) The design process "combines slow reflection with intense periods of very rapid mental activity as the designer tries to keep in mind many things at once." [9]. Hence in collaborative design processes, both asynchronous and synchronous editing phases occur: phases in which individuals work better without the involvement of others, and phases in which problem solving requires intensive discussion. One switches back and forth between these phases as the situation and requirements dictate.
(6) While it is obvious that teamwork in the design process does not mean that everyone is responsible for everything, it should be noted that participants nevertheless influence each other's workspaces. Because the subsystems within the design process overlap, sometimes in many areas, it is not possible to clearly define areas of responsibility for particular aspects of a design.

In the following, we present an overview of the current state of research into the technical basis for distributed collaboration. The methods and systems discussed are evaluated with respect to the aforementioned requirements.

Distributed systems in the context of building design processes

Spatially distributed collaboration on a project requires the use of uniform data standards and effective techniques for data management. At present, the file-based exchange of data and the manual management of versions and variants still dominates planning practice [10]. While the manual management of data is subject to redundancies and errors, data repositories such as Subversion[1] (used, for example, by Gehry Technologies' "Digital Project") or Product Data Management (PDM) can make data management more efficient.

Where different applications are employed simultaneously on one data model, the resulting workflow is non-linear: the applications have both read and write access. This process also generally takes place in parallel on different domain specific aspects, and as a result requires some form of data coordination. Currently model-servers offer the best way to support parallel working processes. These employ an object-based approach and permit the querying, updating and transfer of specific objects in a shared digital model. Current examples of model servers include the Open Source BIM Server[2], the Jotne EPM EDMserver[3] or the Eurostep BIM Collaboration Hub[4].

The Industry Foundation Classes (IFC) serve as a standardised schema for building object models and form the basis of such model servers [11]. An IFC-file created from a BIM application is uploaded to a central server and is then stored object-based in a relational database. From this database, the specific model content can be extracted, again as an IFC-file, for further editing using other applications. When the file is re-uploaded to the model server it organizes the merging of data to produce a consistent model state. The merging of partial models in the model server requires versioning and modification management at the object level [12]. For this the IFC schema provides a mechanism called IfcOwnerHistory which stores the creation date of objects together with a reference to the original application, including the alteration date and the application used[5].

A product of this functionality means that model servers only support asynchronous collaboration between different applications. Furthermore

[1] http://subversion.apache.org/ (last visited 10.01.2011)

[2] http://code.google.com/p/bimserver/ (last visited 10.01.2011)

[3] http://www.epmtech.jotne.com/ (last visited 10.01.2011)

[4] http://www.eurostep.com/global/solutions/bim-collaboration-hub.aspx (last visited 10.01.2011)

[5] *IFC2x Edition 3 Technical Corrigendum 1 (http://www.iai-tech.org/ifc/IFC2x3/TC1/html/index.htm)* (last visited 10.01.2011)

the need to use a generally accepted common data exchange format such as the IFC also means that application-specific building model data has to be translated into the IFC format (see fig. 1). This translation is problematic, because the consistency of the data is not always maintained and the IFC specification does not accommodate certain kinds of model content (e.g. redlining functionality and comments).

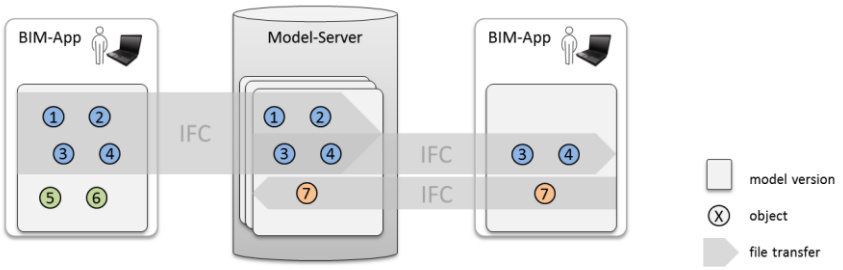

Fig. 1 Model Server principle: Model parts are transferred to the Server via IFC. Only content can be transferred which is supported by the file format.

Some BIM applications provide their own model server functionality by directly managing a proprietary data format. An example is the ArchiCad teamwork function based on the Graphisoft BIM server[6]. Here teamwork projects are stored centrally in the BIM server. At the same time, the project is stored locally on the client computer to facilitate quick responsiveness. If changes are made locally, they must be sent to the server manually and modifications to the central model must likewise be retrieved manually. To edit project parts these have to be reserved by each client. As a result, they are blocked for all the other team members. Changes to a model made by a third party are only visible locally after manually updating the local model. Communication between team members takes place via an integrated instant messaging system. The Graphisoft BIM-Server does not support working with different versions and alternative variants. Because project parts are locked and reserved during editing, this also conditions a linear editing process. Parallel editing of the same project area and with it the development of alternatives is not supported.

For synchronous project work, Autodesk provides a free service called AutoCad WS[7]. This allows the simultaneous editing of CAD data by multiple users over the internet. The files are uploaded to a web server and

[6] www.graphisoftus.com (last visited 10.01.2011)
[7] www.autocadws.com (last visited 10.01.2011)

edited online in real time using a web browser. If several users work at the same time changes are visible immediately. The synchronous editing of drawing elements is possible for each user, and the reserving or blocking of elements is not provided. Previous file revisions can be retrieved using a timeline function. However, the process remains linear and the creation of variants and alternatives is not supported.

A different approach to distributed real-time editing using multiple applications is the project Uni-Verse[8]. The basis for this is the network protocol "Verse" for the distribution of 3D data between different applications. Different applications can communicate via plugins with the verse server, which passes the changes to all clients in real time. Currently plugins for 3ds Max, Blender and Gimp are available. A version management protocol is not supported.

Evaluation of the systems

A consideration of the existing systems and methods for spatially distributed collaboration reveals large discrepancies between the current situation and the requirements of an ideal system (see first section). The most striking point to mention here is the linearity of the systems. None of the systems mentioned allows parallel editing. Instead, "working together" is simulated, for example by reserving certain parts of the design. These parts are then only available to a single actor until released for general availability.

As described earlier, design tasks are usually very complex, and the corresponding processes are fundamentally non-linear in character. This means that, for example, different parts of a design task overlap creating a variety of dependencies, or that detailing problems require solutions at higher levels of abstraction. This complex interplay can only be overcome by effective collaboration. A key limitation is the sequential storage of design changes (transactions) used by the systems described above, which inevitably leads to a linearization of the editing process. The creative process of "working-together" gives way to machine-conditioned "working in sequence". The creative workflow and process of dealing with actual design problems is interrupted by the system's need to establish who will work first, who will follow, and who is editing what.

If collaborative design is to be properly supported by digital design systems, new non-linear approaches are needed. These approaches must

[8] www.uni-verse.org (last visited 10.01.2011, unfortunately the project is considered to be complete and is not developed any further)

take into account that designing is not an a priori organized process, but rather open and network-like in nature. Nevertheless, to implement such models the core problem lies at a deeper technical level: the exchange and management of data. In the following we present a concept for a technical framework that offers a maximum degree of flexibility for use in collaborative design processes.

Technical Framework

For the development of new software tools we use the self developed framework FREAC (Framework for Enhancing Research in Architectural Design and Communication) [13]. The core of FREAC is formed by a dynamically changeable and extensible product model. Product models are an abstract definition of the physical and functional aspects of a product (as defined in ISO 10303). For modelling they employ an object-oriented paradigm, consisting of classes and their instantiated objects. A dynamic model requires that certain structures must be expanded or modified. These modifications involve, for example, adding and changing classes, attributes and methods as well as objects. In FREAC different aspects (such as geometry, building information, freehand sketches) are organized into sub-models which can refer to each other. In addition to the flexible addition of new sub-models, it is possible to adapt and expand existing ones at run-time (see fig. 2).

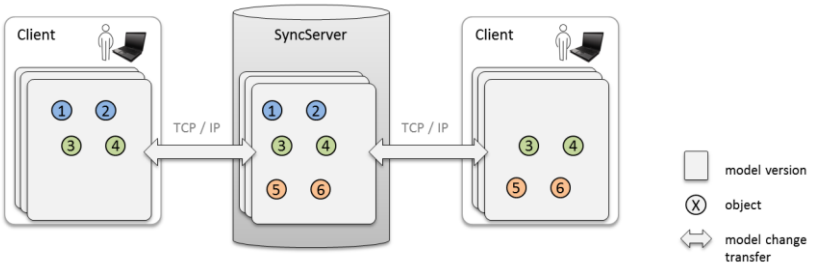

Fig. 2 The SyncServer transfers model changes in real time to all the connected Clients. Accessing Versions is possible from the Clients themselves.

The basis of the framework is the library "DOMMC.Net". This provides all features for the versioned, persistent storage and distribution of models that are built on it. FREAC uses techniques such as transactions and versioning with online synchronization, which allows the storage of the

history of the models as well as their quasi-parallel editing. On a local computer a FREAC-model is edited using applications (clients) for specific tasks. The model data is stored on a central server which transfers model changes to all connected clients in real time and simultaneously manages versioning automatically. Using this client-server-concept, it is possible to fluidly cross-link different clients that can be developed independently of one another (see fig. 3).

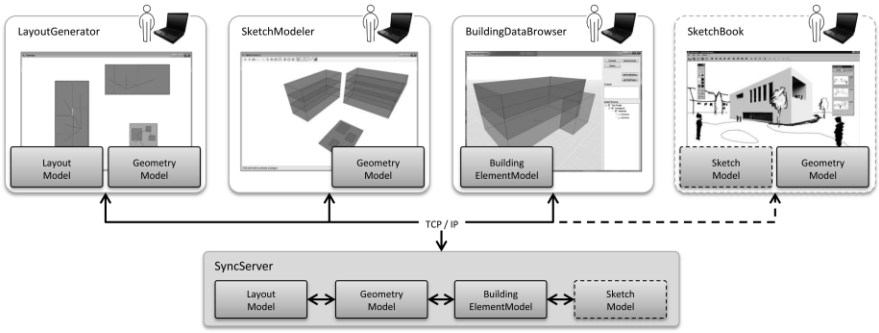

Fig. 3 Currently implemented partial-models and the interaction between Clients and SyncServer. (dashed lines: new submodels, like a freehand-sketching module can be added)

A general versioning concept

The use and creation of tools for design processes always necessitates an intervention in, or a manipulation of the supported process itself. In the case of tools for supporting design collaboration the coordination mechanisms employed represent one of these influence factors. To make these mechanisms as flexible as possible, the challenge at a technical level lies in creating a versioning concept that is as universal as possible.

Within the FREAC framework, model changes are expressed as the creation, deletion or modification of objects. A model change must always transfer the model content from one consistent state into a new consistent state. This change is regarded as a transaction. In the following a consistent model state is called a version. Here too, versions are transformed into new versions through transactions. The versions, or the state of the objects of a model, are stored persistently, which means that the whole history of the model is available. The data volume is reduced by only recording the changes to the model with each version.

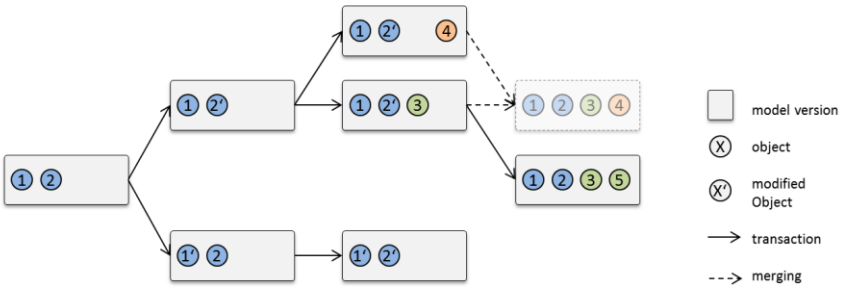

Fig. 4 Versioning through transactions

One version can thereby have many direct predecessor and successor versions. If one version has more than one direct successor versions, these are called alternatives or version branches. The joining of direct predecessors to one version is called merging (see fig. 4). Merging involves a variety of problems, which are described briefly in the next section. To avoid merging during synchronous work as much as possible the parallelization of the editing process is handled pseudo-parallel on the basis of short transactions, in an approach comparable to that used by a multi-tasking single processor. Short transactions are transactions with execution times that the user does not perceive as a delay. The synchronization of the parallel operating clients is effected using a simple token approach.

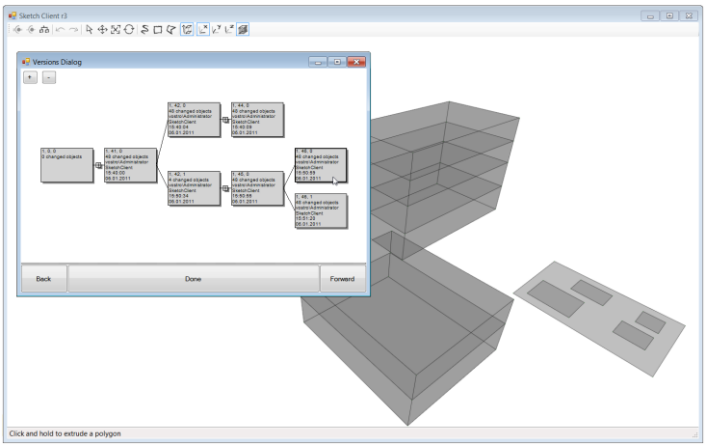

Fig. 5 Currently implemented user-dialog for accessing versions in a FREAC-Client

In order to reduce network load and to increase the response time during object requests, remote method calls have deliberately been avoided. Instead the object content is synchronized with all clients after the completion of a transaction. Accordingly, method calls always take place on the local computer. This approach is optimal where, during the transactions, only a small number of objects are changed. Here the number of objects changed has a greater impact than the number of changes to a single object. In addition to the persistent storage of object content within the versions, additional information such as the author, editing time and the performed operation are also assigned. The latter items in particular, play an important role for automatic merging.

Automatic merging

Merging is a technological process familiar from the context of software development. Source files are modified by different developers in parallel and subsequently joined together. One can differentiate between two approaches: the pessimistic and the optimistic one. With the pessimistic approach, each developer is assigned a specific part of the source code, and changes are blocked for all the other developers. The optimistic approach by contrast assumes that changes by different developers will usually not collide. Where a collision occurs, the system first attempts to merge automatically. If this does not succeed, merging is undertaken manually.

In our case, we transfer this second approach to the level of the object-oriented paradigm (OOP): objects are changed instead of files. Unlike source code file-merging, object content (in the sense of OOP) is virtually impossible to merge manually. Automatic merging is, however, also not trivial. It is difficult, sometimes impossible, to simply derive objects from the changes of the contents of the merged objects. Instead, the algorithms must be described through the operations conducted in the previous versions and the user can then resolve contradictions between operations. It is therefore not enough to merely store the content of an object; it is also necessary to store the operations performed during the transactions.

A matter of negotiation

Automatically merging different versions of a design is a very challenging technical problem. We have discussed some approaches to solving this problem in the previous section. Nevertheless, one should also be aware that merging is not only a technical problem but also a conceptual one grounded in the nature of design problems themselves. One characteristic of these 'wicked problems', as Rittel and Webber [4] call them, is that defining the problem itself inherently also determines the nature of the solution. This definition, however, depends on the cognition, the level of knowledge and the experience of the designer. Accordingly, defining the problem is to a certain degree also arbitrary! Design decisions are therefore inevitably subjective by nature. Only through the communication and interaction of different actors can such decisions be objectified. In this respect, objectified does not mean more correct, but rather the reaching of a consensus among participants, which is less subjective than an individual decision.

With regard to the merging of design alternatives, it becomes clear that such automation is inhibited by sometimes quite elementary questions. By way of example: it could transpire that one participant decides to move a window to the left to achieve a well-balanced facade, while another shifts the same window to the right to optimize the interior light distribution. Here there is an obvious conflict and it is likewise clear that the solution does not lie in positioning the window in the middle of the two transactions. It is not possible to automatically merge these two design alternatives.

In such cases an agreement can only be reached via human to human negotiation. In such circumstances, the only thing the versioning system can do is to signal that there are conflicts, respectively to signal that there is a need for negotiation. Exactly how such a signalling-system should be designed is a matter for future research. Possible points of reference can be found in the concept of boundary objects [14], familiar from the field of Computer Supported Collaborative Work (CSCW). In our context, the digital model represents a boundary object that is formed, interpreted and developed during the design process. In addition to its manifestation through visible artefacts (models, drawings, sketches, text), this boundary object also consists of all options and alternatives that have arisen during the editing process. Each state of an object has predecessors and successors and therefore contains information about how it was created, and how it was developed further. This information can provide useful knowledge about the object being designed and push forward the

negotiation process (discussions, emergence of groups of interest, interaction, brainstorming).

Conclusion

Versioning is a key issue in computer-supported collaborative design processes. For this to be more than just one more item on the feature list of a CAAD system, it is essential to take into account criteria such as the characteristics of design in the conception of the versioning system. These concern, for example, ways of working on a shared model, the coupling of different heterogeneous tools (applications) with the model, the possibility of asynchronous as well as synchronous collaboration, and the non-linear recording of the entire editing process.

The versioning approach proposed illustrates at a technical level, a general solution for the versioning of digital objects (design artefacts). This universality is conditioned by the diversity of cases that arise when designing and also forms a flexible foundation for further work.

In addition to the general versioning approach at a data level, we have also discussed the problems of automatically merging versions. We have briefly elaborated a possible solution via a hybrid approach consisting of object- and operation-based versioning and also highlighted the problems of merging at a conceptual level, and the inevitable need for human to human negotiation in their resolution.

Future research will be concerned with supporting this negotiation process through coordination mechanisms that make use of the proposed versioning concept. Issues include aspects such as navigation between versions, similarities between different alternatives and how these can be used for structuring the emerging diversity, interaction and the handling of the exchange of information in a complex network of versions, the transfer of elements from one version to another and their coordination, comparison or evaluation. Last but not least, we would like to ascertain to what degree it is possible to extract design intentions from the tracking of the process.

References

1. Latour, B. and A. Yaneva, *"Give me a gun and I will make all buildings move": An ANT's view of Architecture'* in *Explorations*

in Architecture: teaching, Design, Research, R. Geiser, Editor. 2008, Birkhäuser: Basel. p. 80 - 89.
2. Maher, M., Z. Bilda, and L. GÜL, *Impact of Collaborative Virtual Environments on Design Behaviour*, in *Design Computing and Cognition '06*, J.S. Gero, Editor. 2006, Springer Netherlands. p. 305-321-321.
3. Sebastian, R., *Managing Collaborative Design*. 2007: Eburon Academic Publishers.
4. Rittel, H. and M. Webber, *Dilemmas in a general theory of planning.* Policy Sciences, 1973. **4**(2): p. 155-169.
5. Schmidt, K. and I. Wagner, *Ordering Systems: Coordinative Practices and Artifacts in Architectural Design and Planning.* Comput. Supported Coop. Work, 2004. **13**(5-6): p. 349-408.
6. Tellioglu, H., *Keeping artifacts alive: towards a knowledge management system*, in *Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*. 2009, ACM: Ruse, Bulgaria. p. 1-6.
7. Schön, D., *The reflective practitioner: how professionals think in action*. 1983: Basic Books.
8. Arnheim, R., *Visual thinking*. 1969: University of California Press.
9. Lawson, B., *Oracles, draughtsman and agents: the nature of knowledge and creativity in design and the role of IT.* Automation in Construction, 2005. **14**(3): p. 383-391.
10. Benning, P., et al., *Collaboration Processes - A State of the Art*. 2010.
11. Eastman, C., et al., *BIM handbook: a guide to building information modeling for owners, managers, designers, engineers, and contractors*. 2008: Wiley.
12. Nour, M., et al., *Overview of Information Management Applications, Including Object-Based Version Management*. 2010.
13. Koenig, R., Thurow T., Braunes J., Tonn C., Donath D., & Schneider S. *FREAC: A Technical Introduction to a Framework for Enhancing Research in Architectural Design and Communication*. in *Future Cities - 28th eCAADe Conference*. 2010. Zürich.
14. Star, S.L., *The structure of ill-structured solutions: boundary objects and heterogeneous distributed problem solving*, in *Distributed Artificial Intelligence (Vol. 2)*. 1989, Morgan Kaufmann Publishers Inc. p. 37-54.