IMPLICATIONS AND POTENTIAL OF REAL-TIME COLLABORATION FOR THE DESIGN PROCESS

Martin Glas,
Bauhaus Luftfahrt e.V., Lyonel-Feininger-Str. 28, 80807 Munich, Germany

Maximilian Schramme, and **Stephan Krusche**
Technische Universität München, Boltzmannstraße 3, 85748 Garching near Munich, Germany

Keywords: real-time collaboration, system design, tooling, development process enhancement

Abstract

Efficient collaboration among designers is essential for the development process of new concepts and products. Recently, software solutions have become available which support collaboration by allowing distributed editing of documents and data models. Thereby, the involved clients are synchronized in real-time. In contrast to conventional collaborative data management solutions, which use local copies that are occasionally consolidated with a central data repository, the real-time collaboration capability allows designers to work *quasi* on the same data at the same time. Thus, the efficiency of the design process can be improved as information conflicts are avoided and disagreements among developers become apparent as soon as possible.

In this paper we use typical application scenarios and modes of real-time collaboration in aircraft design to define an architecture of extension components for enabling existing tools and process models to leverage the potential of real-time collaboration on the efficiency of the design process. Furthermore, we discuss the implications of the real-time collaboration capability on the individual workflow of designers and the coordination of collaborative design activities and speculate how it influences the willingness of designers to share data and synchronize their work.

1 INTRODUCTION

New products in the aerospace industry become ever more complex as advances in performance stem not only from optimization of components but from an improved integration of these components within the product, and of the product into its economic environment. For instance, more efficient aircraft operation comes not only from more efficient gas turbines but also from a better integration of the engines with the aircraft fuselage. Air-borne transport in general becomes more efficient through better integration with ground-based modes of transport. Figure 1 illustrates how the overall product models constitutes form several models concerned with exemplary aspects of the aircraft. As these aspects are not clearly separated, the aspect models have overlapping content. On the one hand, this overlapping content is necessary to cross-check assumptions and assessments of the future product and to enable the different experts complement each other's work. On the other hand, the different aspect models, such as CAD, FEM, or CFD, are created and refined simultaneously, inevitably leading to inconsistencies. Eventually, inconsistencies are resolved by abandoning a version or opinion of one designer. Effort spent into these abandoned work is essentially lost. Therefore, effective conflict avoidance is crucial for the efficiency of a development process of a new system.

During the design process the product model is mostly realized as a virtual data model which is edited by designers via software tool clients. These clients are part of a collaboration infrastructure which enables distribution of model data, version control, and conflict management. In particular, reliable and secure global data communication networks allow designers to

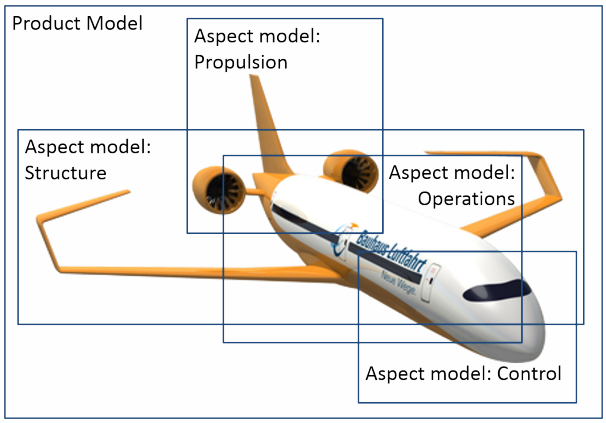

Figure 1: Illustration of different aspect models constituting the overall product model

work at the same time on the same resources but at globally distributed places. In particular, due to low cost and high performance of data transmission the distinction between local and remote data perceived by the user is increasingly driven by the user interface design stipulated by tool developers than by the actual location of data. For instance, a file treated as local resource may be physically located at a remote server.

Currently, most collaboration tool network architectures used in systems design consist of clients and a central server. While the server holds the overall product model in a central repository, each client provides an editor for a local copy of an aspect model relevant to its user. In the following, we describe the *optimistic* and the *pessimistic concurrency control* strategies which both rely on a client-server collaboration network architecture.

As illustrated in Figure 2 an optimistic concurrency control system assumes that most user modifications on shared resources do not interfere with each other [1], e.g., different designers do not edit the same parts of the product model, therefore no conflicts occur. By manual request, designers can merge their local copies with the central copy by loading the latest version from the central repository, resolving conflicts with the local copy if any occur, and upload the merged new version to the central repository. Thus, the optimistic concurrency control system guarantees at any time that the overall product model on the central server is consistent. However, before downloading the latest version from the repository the clients are unaware of any changes, conflicts, and duplicate work between the local versions.

Figure 3 shows that pessimistic concurrency control system grants read and write access to a resource only to one client at a time. Other clients can only read the resource and are prevented from writing by a resource locking mechanism. Thereby, this strategy effectively prevents the existence of two conflicting versions of one resource. However, as designers cannot work on the same resource, this conflict avoidance strategy hinders collaboration on overlapping content [2].

Recently, collaboration systems have become commonly available which synchronize the clients in real-time[1]. As illustrated in Figure 4, changes by one client are instantly propagated to all other clients. Therefore, users effectively work on the same resource. As this strategy completely avoids conflicts between distributed clients, it can significantly improve the efficiency of a collaborative system design process.

We assume that real-time collaboration will be adopted to professional work environments in all industries. However, this paper addresses its application in the aerospace system design process and is structured as follows. In Section 2 we describe typical scenarios of collaboration in system designs. We use these scenarios in Section 3 to define modes and use cases which employ the real-time collaboration capability. To enable these modes and use cases, we define in Section 4 an architecture of components for extending existing system design environments. From the tool developers' perspective, we switch in Section 5 to organizational concerns, where we describe implications of the real-time capability on existing process models.

[1] As described later in Section 4 in detail, these systems only require *soft-real-time* [3] conditions

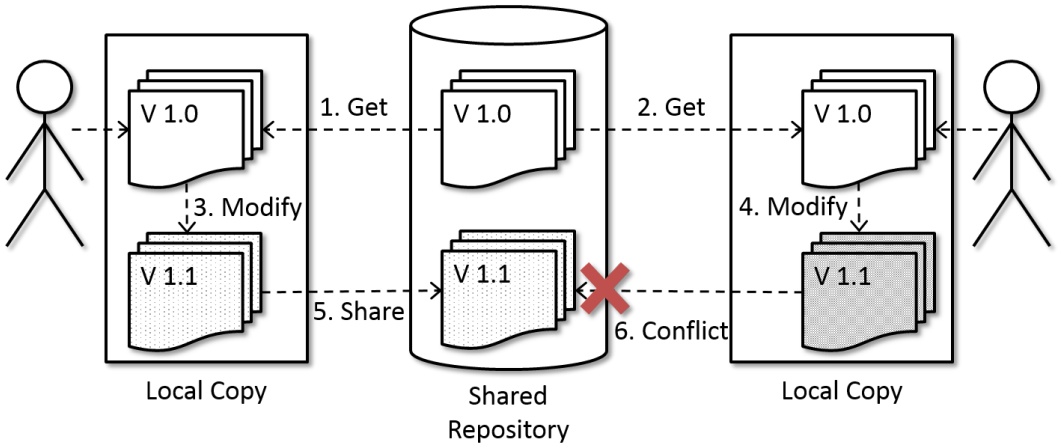

Figure 2: Optimistic concurrency control

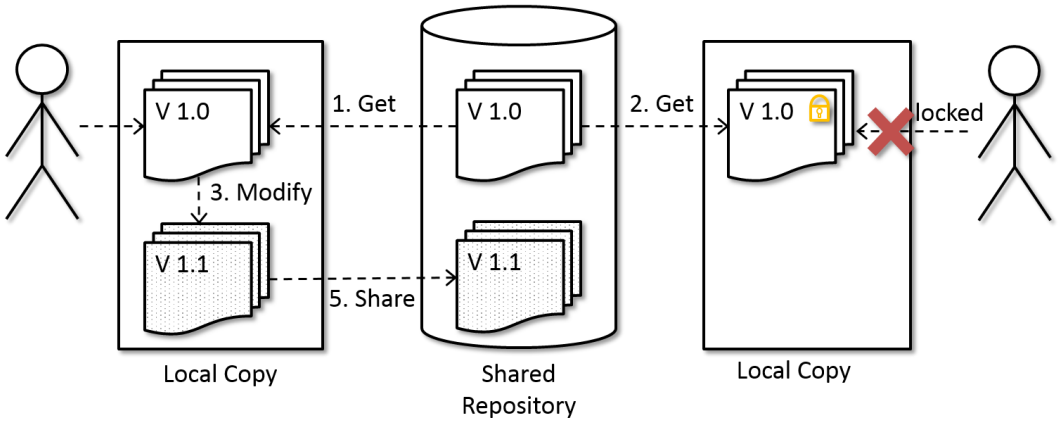

Figure 3: Pessimistic concurrency control

Finally, in Section 6 we discuss the potential of real-time collaboration for altering the motivation of individual designers to proactively share information and to coordinate their work with other designers.

2 COLLABORATION SCENARIOS

In order to provide a concrete context for different modes and use cases of collaboration we describe typical scenarios in a system design process scenarios of systems design where the real-time collaboration capability can become relevant: *Collaborative Meeting*, *Divide and Conquer*, *Back from Vacation*, and *Surrogate versus Numerical Model*. In each scenario we analyze how information sharing and change propagation affects efficiency of its collaborative process.

Collaborative Meeting In this scenario the designers develop a new system architecture using a white board to illustrate and elaborate ideas. At the beginning of the meeting the moderator adds initial content to the white board in order to initiate the discussion by providing a preliminary structure. All participants can add

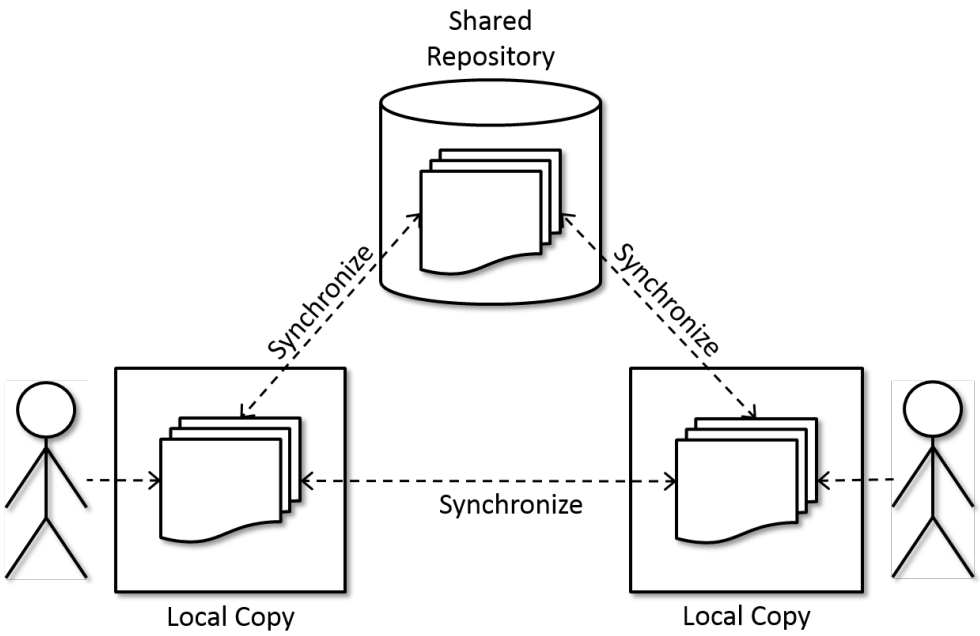

Figure 4: Concurrency control by real-time synchronization of local versions. As there are practically no conflicts between local copies, users are likely to belief to work on the same resource. In contrast to the concurrency control strategy described in Figure 2 and 3 the central shared repository is an optional component.

content which all others can immediately see and comment. Usually wiping the board or parts of it requires the consent of the group. At the end of the meeting the moderator takes a picture of the white board and distributes it to the other participants.

In a collaborative meeting the resource *white board* is shared at the same time and at the same place with all participants which all have about the same authority to edit the content. Due to the limited space on the board, the scopes of all participants are almost identical. Thereby, conflicts or disagreements can be detected and negotiated immediately. Furthermore, due to its limited area, the white board can only hold a limited number of variants of the same concept. Effort spent in creating variants or other content which is wiped from the bard is lost. At the end of the scenario the content of the board is shared as consistent agreed-upon result. However, this result, e.g. the photo of the white board cannot be used directly in the specialized tools of the participants.

Divide and Conquer After a collaborative meeting designers take the results of collaborative meetings, such as images and notes as a basis for working individually on their specific scope of the product model on local tools which are specialized to a specific discipline of design, such as propulsion or geometry. The designers try to avoid duplicate effort or conflicts by dividing the product model into segments which have specified interfaces to each other. The structure of this division of work is based on experiences from previous projects. The designers do not deliberately work at the same time. However, *ad hoc* or regular collaborative meetings take place in order to coordinate efforts, to curate data consistency, or to make design decisions. Thereby,

the designers can use excerpts from their models, such as generated diagrams for substantiating discussions.

In this scenario data conflicts are mitigated by defining interfaces. However, especially in early phases of product development, system boundaries are unclear and overlapping content between different aspect models is intended to facilitate cross-check of assumptions and plausibility of results. However, as the designers have different scopes, they become less aware of changes in the overall product model, thus data conflicts or unintentionally duplicated content is less likely to be discovered. Accordingly, the efficiency of this scenario not only depends on the ability of the team to segment the product model reasonably, but also on its change awareness for the overall product model. To achieve this, the specialized tools used by different experts must maintain a mapping to relevant data between each other in order to automate data transfer between expert tools. Furthermore, there must be efficient data transfer between media used in meetings and tools used for individual work.

Back from Vacation After absence from work a designer must comprehend the new status of the project and the product model. Depending on the extend of overlap with other designers, his scope may have changed significantly. Furthermore, the designer must understand the development which led to the new status. The user must study older versions of the product model and minutes of collaborative meetings. During this phase of regaining awareness for the status of the project the designer also identifies design decisions he disagrees with.

The designer will understand the new status of the project and the product model more efficiently if not only the current status and the status before his absence are available but many status in between and their relation to individual designers and their rationale. In particular, traceable rationale of changes will also facilitate the identification of questionable decisions and disagreements.

Surrogate versus Numerical Model Two groups of designers work together on the assessment of a new propulsion system architecture. The first group is responsible for creating and optimizing the system architecture and uses fast analytical methods or surrogate models. The other group is responsible for the accuracy and precision of performance estimates and uses elaborate numerical models. Due to the different speed of their respective tools, the two groups have different frequencies of producing results. For instance, the product model which was used to start a numerical simulation may have changed significantly during the run of the simulation. Either the latest product model which has been updated by insights gained from faster surrogate models or the result of the numerical simulation is obsolete. In that situation the two groups have to decide whether to integrate both versions by carefully selecting compatible parts from each model or to abandon one version.

The problem addressed by this scenario is product model inconsistency stemming from different process iteration frequencies due to different methodologies. The application of different design methodologies is crucial for the effectiveness of collaborative system design and the soundness of its results. Therefore, different result frequencies cannot be avoided. However, obsolete content can be mitigated by fast propagation of results to all aspects of the product model in order to start a certain methodology with the most up-to-date state of the product model. Furthermore, expected results of a less frequently delivering methodology must be maintained compatible with more frequently delivering methodologies. For example, the group using surrogate models must be notified that certain values in the product models are inputs of a numerical simulation, thus changing them would result in incompatible versions.

3 MODES AND USE CASES OF COLLABORATION

After illustrating collaboration in system design by typical scenarios from an organizational perspective we describe four different modes of collaboration designers can work during these scenarios.

The following modes of collaboration imply that designers use clients for editing parts of the product model. These client have the capability to send changes made by the client to other clients, receive changes from other clients, and merge received changes with the respective local version of the product model. In the scenarios defined in Section 2, a client is usually a software tool but can also be a white board.

Push and Pull Clients sharing resources in this model have to actively push their changes to other clients and pull changes from other clients. Usually there is a central repository where clients send their changes to and, conversely, get up-to-date by pulling shared changes and merging them with the local copy. Both updating and committing changes are manually triggered by the respective user of the client.

This mode of collaboration is provided by common software source code management systems, such as SUBVERSION or GIT. These solutions are often extended by automated change notifications mechanisms. However, the actual changes are not propagated automatically to all clients. MS SHAREPOINT also supports this mode in the "formal co-authoring mode" for general office applications.

Notify Me A Client sharing resource in this mode does not automatically emit changes to other clients. Conversely, the client automatically receives changes emitted by other clients but does not necessarily merge them with its local copy.

Notify Them Clients sharing resources in this mode automatically send local changes to other clients. However, changes emitted by other clients are automatically received but not automatically merged on the local copy.

Real-Time Clients automatically send changes to other clients and merge received changes with their local copies. Thus, resources shared in this mode are synchronized in real-time. This mode is currently supported by general office collaboration solutions, such as GOOGLE DRIVE, MS OFFICE 365, and MS SHAREPOINT in the semi-formal coauthoring mode. These examples, however, so not address applications in system design.

The scenarios of collaboration in systems design described in Section 2 with respect to the modes of collaboration described above imply the following use cases as illustrated in Figure 5:

Switch collaboration mode The user changes the collaboration mode in order to adapt to new a new context of work.

For instance, in the *Collaborative Session* scenario the user wants to share his information in real-time with the participating designers. When the designers are in a *Divide and Conquer* scenario the designers are more likely to switch to the *Notify Them* or *Notify Me* mode, in order to keep their local model stable. As mentioned in the analysis of the *Divide and Conquer* scenario, a seamless change between modes crucial.

Review modification history The user will enter this use case in order to retrace past modifications of shared resources in chronological order. As shown in Figure 5, this use case is closely related to the *Review version differences* which allows the user to compare two different versions of a model.

Both use cases are especially relevant for the *Back from Vacation* scenario where a designer

Mode	Receive	Send	Merge	Examples
Push and Pull	manual	manual	manual	Subversion,Git, MS Sharepoint [2]
Notify Me	auto	manual	manual	
Notify Them	auto	auto	manual	
Real-Time	auto	auto	auto	Google Drive[3], MS Office 365[4], MS Sharepoint

Table 1: Different modes of collaboration on digital resources discriminated by their behaviour in receiving, sending, and merging changes

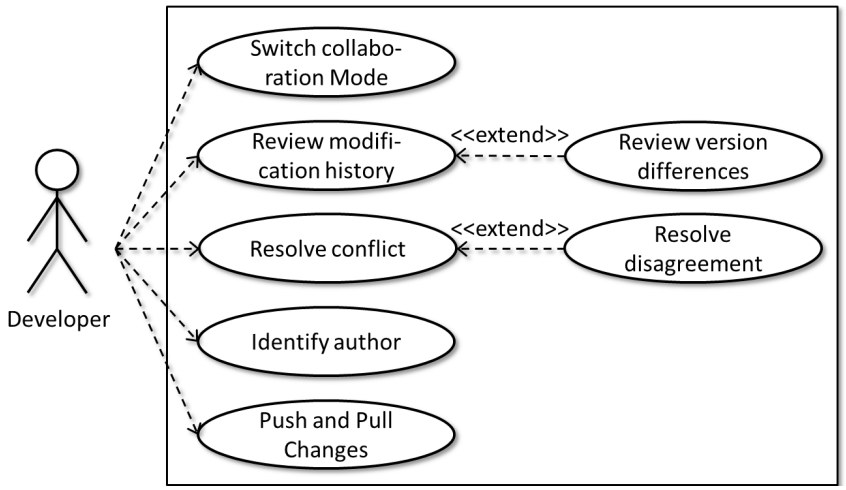

Figure 5: UML use case diagram describing the use cases derived from the scenarios

has to understand changes of the project and product model committed during his absence.

Resolve conflict Conflicts are pieces of information which cannot be true at the same time in the same model. Usually conflicts can be detected automatically. However, conflicts cannot be resolved automatically but require designer's decisions to select one of the conflicting pieces of information. By contrast, disagreements are determined by the opinion of an individual designer and cannot be detected automatically, thus can only be detected after a particular designer becomes aware that an information of interest was changed by another user. While *Resolve conflict* can be performed by individual designers, especially *Resolve disagreement* requires negotiations between designers.

In all scenarios described in Section 2 designers will try to mitigate both use cases by reasonable segmentation of the product model and by sending and merging received changes as soon as possible.

Identify author In a collaborative process, the author or modifier of a particular model element is relevant if a designer cannot resolve a conflict on his own or wants to raise a disagreement. Furthermore, designers assess a change by the reputation of its author.

This use case is especially relevant if designers do not work at the same place or the same time when changes by other designers are not in their

scope. Especially in the *Back from Vacation* scenario the designer will often enter this use case in order to understand the rationale of changes and to find relevant stakeholders for negotiating disagreements.

Push and pull changes As mentioned before the *Push and Pull* mode of collaboration requires manual trigger by the user. This use case typically occurs when the designer regularly updates his local copy with changes from the central repository or irregularly if the designer has finished a work item, and wants to share his modifications. Usually, the designer pushes his changes to the central repository only in the latter case.

4 ENABLERS OF REAL-TIME COLLABORATION

In Section 3 we described different modes of collaboration which are enabled by the capability of tool clients to propagate changes of the product model to all clients in real-time. When looking at current solutions for collaborative editing which support the described modes of collaboration we observe that these solutions are more designed for general office applications than specialized to systems design, not to mention for the specific needs of a certain engineering discipline. However, we pointed out that the propagation of changes between different models with unavoidable overlapping content and a seamless switch between different modes of collaboration are crucial for the efficiency of the collaborative design process. Therefore, in this section we describe an architecture of components depicted in Figure 6. This architecture is supposed to extend existing tools in order to enable the modes and use cases of collaboration.

Change Tracker As a basic capability for real-time collaboration, each client must be able to capture changes on its local copy by its user and provide them as a stream of change notifications. This stream must be fast and meaningful enough that another receiving clients can replicate the changes in real-time.

Editors usually receive and execute user requests on the client model using a combination of the *Command* and the *Observer* pattern [4]. This architecture is usually provided as an application programming interface of the design tool to third party developers. Accordingly, a change recorder extension can "subscribe" to change events and provide them to other real-time collaboration system components.

Version Control Distributed changes to the product model must be managed in a consistent version control system which basically assigns new changes to a globally unique version and specifies their relation to previous changes. Furthermore, the `Version Control` component manages different development branches of a resource.

This component is important for determining the most up-to-date version of a product model in the *Back from Vacation* scenario, for merging complex version differences as in the *Push and Pull* mode or executing fast automated merging in the *Real-Time* mode.

Change Communication A real-time collaboration system must transmit and process the changes between all clients fast enough that the delay by the synchronization cannot be recognized by the user. For the *Notify Me*, *Notify Them*, and *Real-Time* mode we do not require hard real-time but soft real-time conditions as described by Kopetz [3]. For the *Push and Pull* mode the conditions are even more relaxed.

Especially the *Divide and Conquer* requires a seamless change between the different modes of collaboration. Therefore, the `Change Communication` component must support switching between manual and automated sending and receiving. The *Back from Vacation* scenario also requires that the `Change`

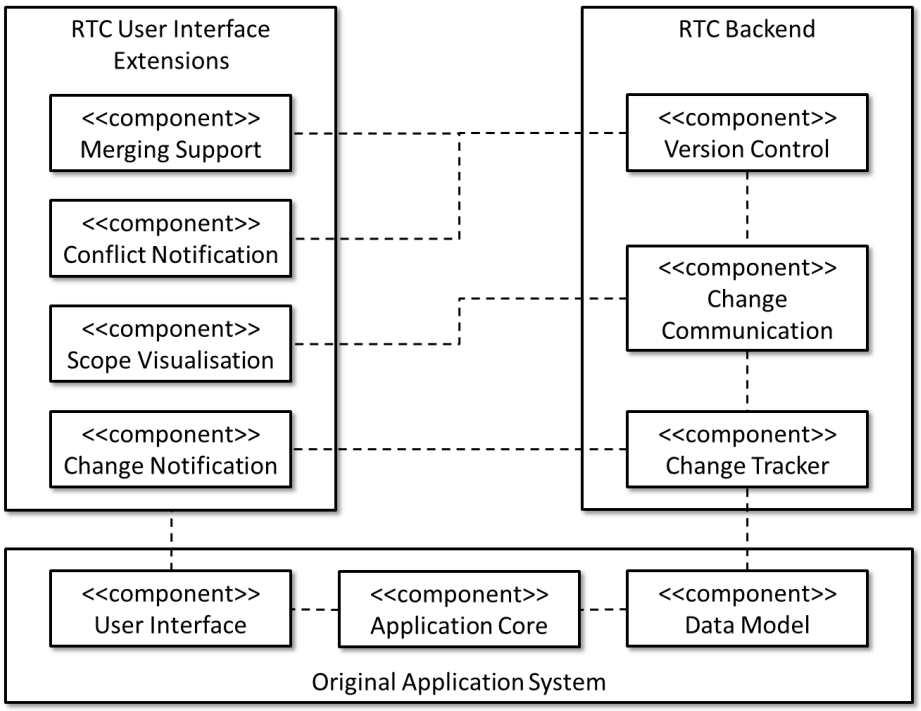

Figure 6: UML component diagram displaying the structure of an RTC extension for an existing application.

`Communication` component must request the complete change history which has been missed.

Scope Visualization This user interface component visualizes the scope of other designers. For instance, if a designer works on the geometric shape of the wing, the affected model elements are also highlighted in the FEM and the CDF model of the wing in other client.

This component is especially relevant in the *Divide and Conquer* and the *Surrogate versus Numerical Model* scenario when the designers have to be aware of specific models parts which are affected by other designers in order to coordinate design activities.

Change Notification An important component to convey change awareness is a display of changes and conflicts. If a user creates or modifies a particular model element and sends change notifications as a stream, other clients receiving this stream will highlight this element indicating the focus of the other user. For example, if a designer is editing a text document in *Real-Time* mode, this component will display his cursor in other receiving clients. The `Change Notification` component must also support individual designers in determining *disagreements*.

Conflict Notification If a designer chooses to merge incoming changes manually, this user interface component indicates model elements in conflict with incoming changes. The notification must support the designer in assessing the complexity of the conflict.

The user interface components `Scope Visualization`, `Change Visualization`, and `Conflict Notification` are important for the process efficiencies of the *Divide and Conquer* and the *Surrogate versus Numerical Model*

scenarios when the designers have to be aware of specific models parts which are affected by other designers in order to coordinate design activities.

Merging Support If conflicts or disagreements occur, the affected designers must be supported in resolving the conflict by selecting the best pieces of information from both sides of the conflicting versions in order to avoid loss of valuable content and effort which went into creating the content.

This component is only relevant if designers have chosen to merge incoming changes manually. Otherwise, incoming changes are merged by a simple automated strategy. For instance. in Google Drive, the change message with the latest time stamp supersedes all older change messages.

5 IMPLICATION FOR PROCESS MODELS

We define a process model as a sequence of activities, roles, and artifacts which are assembled to a project structure by applying certain rules. A group of designers applies a process model to give their work process a structure in time which facilitates coordination of progress and managing the risk of quality, time, and costs.

The availability of different modes of real-time collaboration affects the effectiveness and efficiency of process models when aiming at efficiency, robustness, situation awareness, controllability, and traceability:

Efficiency Developing a new product utilizes limited resources, such as machines, personnel, time, and capital. A process model provides guidelines for prioritizing resource budgets and controlling their allocation.

Collaboration using real-time synchronization can improve efficiency as conflicts are detected earlier, thus conflict resolution by selecting the best combination of conflicting pieces of information is more realistic, thus less work becomes obsolete. In particular, the *Real-Time* mode completely avoids information conflicts. Furthermore, all modes described in Section 3 are optimistic concurrency control strategies which allow all designers to edit the shared resources at the same time.

Process models can foster or prescribe a certain policy of using real-time synchronization or the other modes of sharing in order to support designers in exploiting these potentials.

For instance, a process model could prescribe that in *Collaborative Meetings* all participants should use the *Real-Time* mode whenever possible and *Notify Them* mode in *Divide and Conquer* phases.

Robustness Every development process faces unanticipated external and internal disturbances. A process model must support the project participants to recover from disturbances and stabilize the development process and the process model. Conversely, the process model should enable the participants to react on disturbances adequately and realize learning effects.

The the fast propagation of changes by automated merge of changes can lead to instability or even disintegration of the product model if circle dependencies between model elements do not converge. Conversely, in the *Surrogate vs Numerical Model* scenarios we described different frequencies of result delivery as an intrinsic reason for product model instability, which can be mitigated by real-time collaboration. Design process models should give advice on how to mitigate and confine unstable model segments, for instance by temporally locking certain model elements.

Situation Awareness and Controllability The process model must support the participants of the project, in particular managers, in

perceiving the status of the project and steering it towards a potentially changing goal.

In a previous publication [5] we argued that pushed notifications about changes of the project schedule or the product model can enhance situation awareness of project participants. In that sense, the *Notify Me*, *Notify Them*, and *Real-Time* modes contribute to general situation awareness by pushing change notifications automatically. We then also proposed how the abundance of change notifications could be organized by boundaries and interfaces which may be aligned with the structure of the product or the organization.

We claim that a system which propagates changes at a wide range of speeds has a considerable more chaotic system response to control actions than a system which propagates with almost no delay. Accordingly, a process model can contribute to the controllability of the system design process if it fosters a broad adoption of real-time collaboration modes among designers.

Traceability Especially for process models addressing the development of safety critical systems traceability is a paramount concern. Real-time collaboration can contribute to traceability, e.g., clients in the modes *Notify Them* and *Real-Time* mode may allow tracing every change to an individual designer. However, especially due to the catastrophic failure modes of safety critical systems, designers may be reluctant to share changes for which they might be made accountable. Therefore, process models must define policies for using this information without unfairly violating privacy.

6 POTENTIALS OF REAL-TIME COLLABORATION

In the two previous sections we described the technical and organizational implications of real-time collaboration for the design process. In this section we approach the question about the potentials of real-time collaboration form the individual designer's perspective.

For a designer the most essential actions of collaboration are sharing content and working with other designers at the same time on those shared resources. Only with shared content collaborative design systems and methodologies are effective. Furthermore, despite the complex coordination, the collective of designers, profit the most from collaboration when working at the same time on the same resources mainly for two reasons. First, conflicts disagreements become apparent almost instantly with all stakeholders present for negotiations. Second, designers can support each other's work by giving direct expert feedback and providing complementary content.

Despite the advantages for the collective, individual designers may experience disadvantages which diminish their intrinsic motivation for collaboration. First, sharing content requires additional effort, such as documentation and standardization of the content. Furthermore, especially when sharing content in early phases of its development, a designer exposes his own immature and experimental work, which he might fear to be accountable for later.

Therefore, we assess the potential of real-time collaboration in how it might affect the motivation of designers to share more content earlier, and to coordinate their work towards editing the same content at the same time. Therefore, we speculate which intrinsic motivation a designer might have to use a specific mode of collaboration:

In the *Push and Pull* mode the designer has most control about sharing changes and on his local version of the product model. So when he initially joins a collaborative circle of designers and decides to share content, this mode appears to be the safest choice. However, conflicts and disagreements are detected not before a manual pull of remote changes. Accordingly, the *Push and Pull* mode potentially leads to more compli-

cated conflicts and disagreements, thus obsolete content.

When using the *Notify Me* mode the designer, has full control over his local version of the product model, but becomes aware of potentially conflicting or disagreeable changes, as they occur. Therefore, this mode diminishes the risk of complicated conflicts and disagreements by improving the designer's situation awareness for project and its product model. However, the postponed sending of local changes might lead to a complicated package of changes. In order to ensure the stability of the overall product model, other designers working in the *Real-Time* mode might demand a pre-merge review, which might be similar to the merging effort required in the *Push and Pull* mode.

By using the *Notify Them* mode the designer exposes his work directly to other designers. On the one hand, this might improve his working efficiency, as he avoids conflict resolution while keeping his local version isolated from external changes. On the other hand, the full transparency of his work might prevent him from elaborating immature or even radical concepts.

The *Real-Time* mode gives up control over his local copy, by allowing auto merge of remote changes. However, this mode provides most control over the overall product model. Other designer might recognize the product model shared in *Real-Time* as the most recent agreed-upon version, and treat it as a reference. Furthermore, two or more designers working on the same scope in *Real-Time* mode can share information without the overhead of explanation and documentation as they share the same context. Thus, their interaction becomes more efficient.

All in all, we argue that especially the *Notify Me*, *Notify Them*, and *Real-Time* modes which make use of the real-time collaboration capability, will provide direct benefits for the individual work efficiency. It is even imaginable, that a designer might decide to leave the *Real-Time* mode only in rare periods in order to avoid conflict resolution. With increased change awareness, the individual designer might also be motivated to synchronize working periods on the same scope of the product model with other designers in order to mitigate late disagreements. However, a real-time collaboration solution for system design must address product model instability and the designer's fear of negative exposure both by technology and policy in order to ensure sustainable application. For instance, the design editors, should allow to run parts of the product models in different modes and to confine direct change propagation within defined system and organizational boundaries.

7 RELATED WORK

There is a great body of literature on computer supported collaborative work (CSCW). Dourish and Bellotti [6] investigated awareness in shared work spaces. In particular, they noted that a shared context is crucial for the direct benefit of an individual sharing information to a group.

Carstensen and Schmidt [8] highlight the complexity of collaborative design compared to other fields of cooperative work, such as process control and manufacturing. They define three challenges for CSCW-systems design and research: A better understanding of the "natural attitude" of individual actors in a collaborative context, a better understanding of how to achieve awareness among collaborative designers, with respect to different views on design, and building basic platforms and components in order to realize flexible solutions for collaborative design.

Schmidt and Bannon [9] emphasize that CSCW-systems should understand the nature and requirements of cooperative work. They state that the interaction and dependency of multiple individuals have important implications for the design of collaborative systems and describe articulation work, the partition of work into units, as one particular issue focusing on supporting the management of workflows.

Strauss [10] writes about the division of labor and its impacts in projects, while focusing on rapidly changing organizations and industries. They describe two important types of work patterns, the closely collaborative and the harshly conflictful, and discuss certain conditions and their implications for aspects of the division of labor, e.g. the rate of task structure change.

8 CONCLUSION AND OUTLOOK

We described the key components of a real-time enabled collaboration system and typical user scenarios. We also proposed an architecture for extending existing system design tools with real-time collaboration capability. We have been developing a real-time collaboration extension framework [11] currently focusing on the `Change Communication` component as depicted in Figure 6. A prototype of this framework has been integrated in our experimental open source system design platform OpenCDT[5].

We also showed how design process models can be improved by integrating real-time collaboration in their methodology. We are currently working on a system which simulates real-time collaboration as a sociotechnical system. Despite its simplification of the collaborative design tool network and the behavior of the developers, the simulation allows us to test hypotheses of designers' behavior and evaluate our extension framework.

Finally, we argued that real-time collaboration can improve the intrinsic motivation of individual designers towards sharing more content earlier and to work more frequently with other designers on the same scope of the product model. Accordingly, real-time collaboration has the potential to improve the work efficiency not only of the collective but also of the individual designers.

[5] `http://www.opencdt.org/` last access 2.9.2014

REFERENCES

Kung, H. T. and J. T. Robinson (1981). "On Optimistic Methods for Concurrency Control". In: *ACM Trans. Database Syst.* 6.2, pp. 213–226.

Song, X. and J. W. .-.-S. Liu (1995). "Maintaining temporal consistency: pessimistic vs. optimistic concurrency control". In: *Knowledge and Data Engineering, IEEE Transactions on* 7.5, pp. 786–796.

Kopetz, H. (2011). *Real-time systems: design principles for distributed embedded applications*. Springer.

Buschmann, F. (1999). *Pattern oriented software architecture: a system of patters*. Ashish Raut.

Glas, M. and S. Ziemer (2009). "Challenges for Agile Development of Large Systems in the Aviation Industry". In: *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*. OOPSLA '09. Orlando, Florida, USA: ACM, pp. 901–908.

Dourish, P. and V. Bellotti (1992). "Awareness and Coordination in Shared Workspaces". In: *Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work*. CSCW '92. Toronto, Ontario, Canada: ACM, pp. 107–114.

Johansen, R. (1988). *GroupWare: Computer Support for Business Teams*. New York, NY, USA: The Free Press.

Carstensen, P. H. and K. Schmidt (1999). "Computer supported cooperative work: New challenges to systems design". In: *In K. Itoh (Ed.), Handbook of Human Factors*. Citeseer.

Schmidt, K. and L. Bannon (1992). "Taking CSCW seriously". In: *Computer Supported Cooperative Work (CSCW)* 1-2, pp. 7–40.

Strauss, A. (1985). "Work and the Division of Labor". In: *Sociological Quarterly* 26.1. Zuletzt aktualisiert - 2013-02-24, p. 1.

Krusche, S. and B. Brügge (2014). *Model-based Real-time Sysnchronization*.