Integrated Multiagent and Peer-to-Peer based Workflow-Control of Dynamic Networked Co-operations in Structural Design

S. Alda & A. B. Cremers
University of Bonn, Germany

J. Bilek & D. Hartmann
University of Bochum, Germany

ABSTRACT: Modern engineering projects in the application domain of structural design are organized in networked co-operations due to permanently enlarged competition pressure and a high degree of complexity while performing concurrent design activities. One of the major challenges of these networked co-operations constitutes the coordination of the activities of all involved participants. In the course of our common research activities, we have developed two different directions for coordinating these projects: i) a workflow-based concept regulating the activities explicitly by a global workflow model (University of Bochum) and ii) an awareness model that allows to perceive activities of other protagonists and to derive new activities mentally. This paper describes a novel integration approach of these two models: according to the global workflow model, users can be connected through awareness channels that enable them to detect potential inconsistencies during their concurrent modeling activities. Inconsistencies that would otherwise remain in the project progress can thus be discovered at a very early stage.

1 INTRODUCTION

Large engineering projects in construction engineering necessitate the involvement and the close collaboration of different engineers and other specialists residing in different geographic locations. We indicate such virtual project constellations as so-called *networked co-operations*. Networked co-operations in construction engineering exhibit major benefits in particular for organizing complex planning processes. Engineers can be involved in many projects concurrently, while residing in their local offices and, thus, reducing the costs for time-consuming business travels. Modern software systems allow these participating individuals to share project documents (e.g. IFC-based CAD documents), to receive an accumulated view of the overall project progress, and even to execute verification algorithms to partial building models for establishing the consistency to other models.

In the course of our common research activities carried out at the Universities of Bonn and Bochum, we have proposed and developed an integrated multi-agent and peer-to-peer software architecture *(MAS-P2P)* for supporting collaborative structural design processes within networked co-operations. Our software architecture covers fundamental aspects of a component-based peer-to-peer model for the flexible integration of heterogeneous software in a co-operation (project *CoBE*, Bonn) as well as a generalized agent model utilizing collaborative structural design based on an agent-based product and workflow model (project *ACOS*, Bochum).

In the current version of our approach we have implemented two different approaches for *coordinating the activities* of all project members belonging to a networked co-operation. First, an agent-based process support model is available to control and distribute the overall workflow of the project to the appropriate engineers. A workflow agent has been implemented to take over this task. The capabilities provided by the workflow agent are based on the Petri net theory. Based on Petri nets, the workflow agent links design processes to project resources (product models, software, and human experts). Besides, the CoBE awareness model enables human users (e.g. structural engineers/designers) and ACOS software agents to directly perceive planning activities that result from the modification of local (partial) product models. We already indicated that the adoption of software agents into the CoBE awareness model (both systems are technically concatenated by the *consistency agent*) is an important improvement, especially to guarantee the consistency of the distributed partial product models throughout the entire design life cycle.

However, our firstly proposed MAS-P2P architecture makes a couple of rather weak assumptions for the sake of simplification: At first, it premises an existent, already initialized static co-operation net-

work, in which all pertaining users and agents are connected initially. At a given point in time within a project, no assumptions can be made, which of the enrolled users should or *must* actually be connected through CoBE's awareness channels. Also, neither the exact rationale nor the duration of their connection with respect to a global (process) model can be defined for a connection among different people.

In this paper, we propose a further integration of our two coordination models, that is, the awareness model and the agent-based workflow model within the existing MAS-P2P approach. Given a global workflow model being controlled by a workflow agent, project members are connected *dynamically* through an awareness channel by this agent, i) if their activities *are about to be executed* concurrently within the workflow and ii) if their activities refer to the *same dataset*. So, potential inconsistencies that could occur during parallel modeling activities can be detected early enough in this stage of the workflow through the awareness of the single activities.

The rest of this paper is structured as follows: section 2 provides background information about both involved projects including a brief introduction about the inherent coordination models for concurrent design activities. Section 3 presents the MAS-P2P architecture including an extensive description of the integrated coordination model. Section 4 finally concludes this paper.

2 BACKGROUND: ARCHITECTURAL SUPPORT FOR STRUCTURAL DESIGN PROCESSES

In the course of the priority program 1103 "*Networked-based co-operative planning processes in structural engineering*" funded by the German Research Foundation, we have scrutinized possible software architecture models for enhancing networked co-operations within collaborative structural design work. In this context, two major models evidenced as feasible: 1) the *peer-to-peer* architecture model that was evaluated by the University of Bonn and 2) the *multiagent* architecture evaluated by the University of Bochum, respectively. In the subsequent sections, we will first outline fundamental concepts of both models. Hereby, we primarily focus on the coordination between design activities that are carried out by the various structural design experts concurrently. The first, peer-to-peer based research project supports synchronous and asynchronous design activities by implanting the *awareness framework*. In contrast, the second, agent-based research project utilizes *workflow techniques* based on the theory of *coloured Petri nets* for the coordination of the concurrent design activities. Both coordination approaches reveal particular pros and cons that are pointed out at the end of this chapter.

2.1 Multiagent Architecture

Multiagent systems (MAS) are a rapidly developing area of research that emerged from the distributed artificial intelligence (DAI). Within the Bochumer project a *software agent* is defined as an *encapsulated software unit that is situated in a dynamic, heterogeneous environment, capable of solving well defined tasks autonomously and pro-actively in co-operation with other agents by order of personal (human) or non-personal principals.* In practice, software agents primarily assist their assigned design experts in their activities aside the support of other software agents in the environment. Several interacting agents, thence, are understood as a multiagent system. The adaptation of agent technologies to the specific needs and requirements of collaborative structural engineering implicitly requires the *decomposition* of the *entire structural design process* into adequate, domain-specific interacting agents (Bilek & Hartmann 2003).

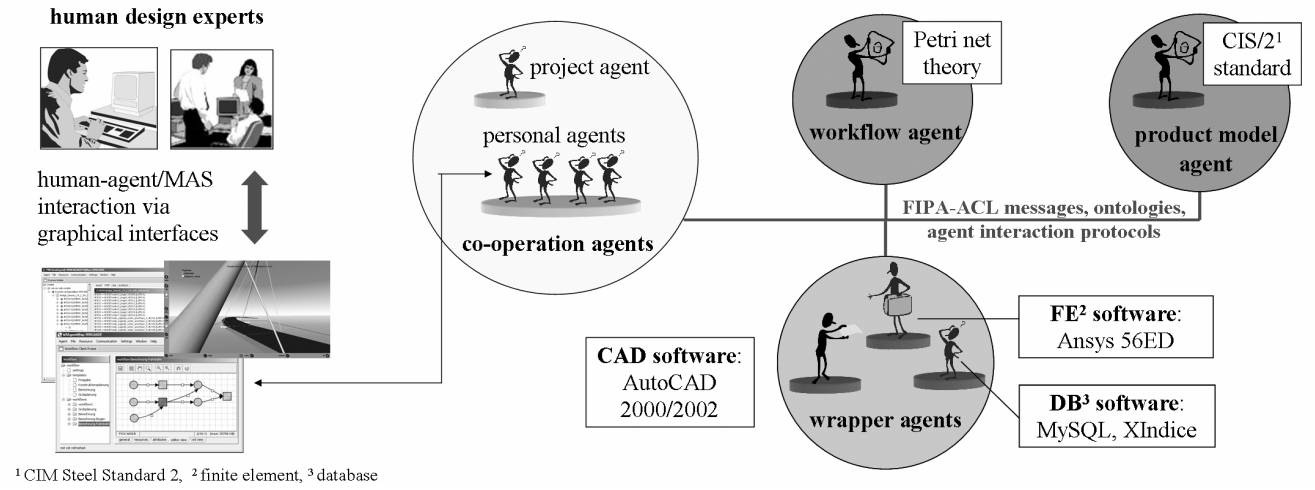
In our research work we identified the following four substantial domains that were incorporated into the overall *agent model for collaborative structural design:* i) the participating specialized design experts and their associated characteristic, dynamic organizational structures (agent-based co-operation model), ii) the specific structural design processes (agent-based process support model), iii) the associated (partial) product models (agent-based product model), and iv) the applied, engineering standard software (agent-based software integration model).

Within the *agent-based co-operation model* (i) human experts and their assigned, design-specific organizations (like specialized engineering offices, other building companies, and authorities) are represented by *co-operation agents*. It is vital to differentiate between two fundamental co-operation agent types, the *personal* and the *non-personal co-operation agents.*

Personal co-operation agents directly assist their assigned individuals in their design activities. For that reason they provide a graphical user interface that enables the human experts to interact with other project associates and the multiagent system.

Non-personal co-operation agents can be seen as virtual representatives for the participating organizational entities, primarily the building companies. They hold specific information about their associated organization like enlisted project co-workers, time schedules and design tasks that have to be conducted. A very special non-personal co-operation agent is the *project agent*. It represents and supports the common, intrinsic project work by supervising and controlling the project handling, administrating the project members, responsibilities and delegating design tasks.

The next fundamental agent-based submodel, the *agent-based process support model* (2), is to control



[1] CIM Steel Standard 2, [2] finite element, [3] database

Figure 1: Agent model for collaborative structural design and prototypical implemented agents within ACOS

and allocate the complex design processes to convenient participating designers. For that, a *workflow agent* has been modelled as a delegate to the project agent. Based on the Petri net theory, the workflow agent links design processes to project resources (product models, software, human experts). For a more detailed depiction of the incorporated Petri net based workflow concepts see section 2.3.

The third basic submodel, the *agent-based product model* (3), rests on the decomposition of the entire structural system into smaller structural subsystems. Each structural subsystem thereby is accessible via an assigned product model agent that owns and stores knowledge about several structural elements created by the participating structural designers during the design process. The product model agents adjust their knowledge and, thus, check structural dependencies and retain product model consistency. The product model specification used conforms to the CIS/2 (CIM steel integration standard) standard. CIS/2 is a set of formal computing specifications that are based on ISO 10303 (STEP – STandard for the Exchange of Product model data) and allow the modeling of steel structures. CIS/2 covers the three major planning phases structural analysis, structural design and fabrication aside data management issues (Reed 2002).

The integration of heterogeneous engineering standard software (such as CAD-, FE-programs, databases) is achieved through the implementation of wrapper agents as specified in the *agent-based software integration model* (4). Wrapper agents operate as interfaces between the MAS and locally installed software such that the software applications can be used by all other agents and/or human design experts in the agent network.

In the course of the prototypical implementation ACOS (**A**gent-based **CO**llaborative **S**tructural Design) the delineated four submodels have been simplified such that only basic and necessary conceptual elements must be implemented into a set of design-specific software agents (Fig. 1). These agents interact with each other by exchanging several speech-act based messages that conform to the FIPA (Foundation for Intelligent Physical Agents) agent communication language (FIPA-ACL) specifications (FIPA 2000). The realisation of complex dialog structures is provided by task specific interaction protocols (IP). In addition, several domain specific ontologies have been developed (product model ontology, workflow ontology, etc.) and concatenated with agent specific capabilities. It has proved that the sum of several lously-coupled, design specific software agents in a dynamic environment is a convenient, flexible and applicable way to support collaborative structural design activities.

2.2 Component-based Peer-to-Peer Architecture

The peer-to-peer architecture concept is a novel abstraction for developing software aimed to support virtual organizations. Following this style, a peer-to-peer architecture constitutes a distributed architecture that consists of uniform clients or so-called peers. Peers are capable not only of consuming, but also of providing computer resources like data, legacy applications, proprietary software solutions, simple routines, or even hardware resources. These resources are encapsulated by *peer services*. In contrast to other service-oriented architectures, peer-to-peer architectures assume an unstable and dynamic topology as an important constraint. That is because peers are solely responsible to affiliate to a network. Beyond the possibility of direct resource sharing, peer-to-peer architectures enable single peers to organize into so-called peer groups. These self-governed communities can share, collaborate, and communicate in their own private web. The purpose is to subdivide peers into groups according to common interests or knowledge independent from any given organizational or network boundaries.

Peer-to-peer architectures have mainly been used to implement file sharing systems such as Gnutella or Napster. The exchange of files, thus, constitutes

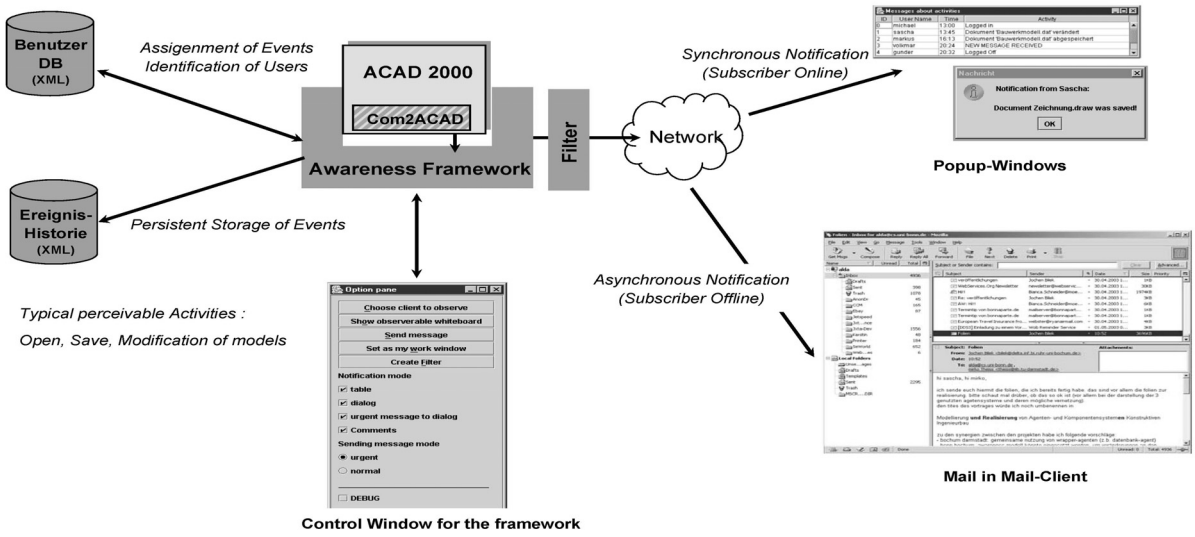


Figure 2: The CoBE awareness model.

the only service that is shared among all participating peers. Other systems such as Seti@Home integrated unused computing cycles towards a virtual super computer that allows the concurrent computation of a massive amount of data. Here, the access to a local processor corresponds to the only shared service. The brief illustration of the above mentioned systems reveals that peer-to-peer architectures principally support the *vertical integration* of services, that is, only a small number of service types are integrated. The de facto standard peer-to-peer framework JXTA leverages the original notion of peer-to-peer towards a rather service-oriented architecture, where a wide range of services can be published and discovered by a single peer. However, the JXTA approach exhibits two apparent drawbacks: first, no notation is provided to describe compositions of services towards higher level application. Secondly, no mechanisms are implemented to react on the unavailability of peer and peer services.

In the course of the research project CoBE, we have developed DeEvolve (Alda and Cremers. 2004), a self-adaptable peer-to-peer architecture based on top of the JXTA standard peer-to-peer framework (Sun 2004). DeEvolve incorporates the component technology as another fundamental technology. According to the component technology, peer services are made up by the composition of single software components. A component model called FlexiBean does prescribe the valid interaction primitives for both the local interaction within a service and the remote interaction among distributed services. Peer services can be made available (published) to other peers by means of advertisements. Each service can be assigned to at least one or more group affiliations, in order to restrict the access to a service for authorized group users. Users of other remote peers are able to discover and use these services. Additionally, we provide a composition language called PeerCAT (Alda and Cremers 2004), which enables users to declare the composition of different peer services towards individual, higher-level applications. As a self-adaptable architecture, DeEvolve is capable of detecting and resolving unanticipated exceptions such as the failure or unavailability of peers. The handling of exceptions is done in strong interaction with users: a user, for instance, is able to decide which routine is to be executed for resolving an occurred exception. DeEvolve is accompanied by a couple of auxiliary tools not only for the discovery, advertisement, and composition of services, but also exception handling, as well as for the management of groups.

2.3 Coordination of concurrent Design Activities

In both research projects different process support models are favored: the agent-based project makes use of high-level *Petri net based workflow management concepts* (van der Aalst 1998) whereas the peer-to-peer based research project avails *awareness based approach*.

The theory of awareness for regulating distributed activities has become a popular research topic in the area of groupware systems. These systems are usually deployed in an environment consisting of a group of people, which perform the general role of supporting the co-operation within this group. According to the author Dourish, awareness is an understanding of the activities of others, which provides a context for your own activity (Dourish 1992). With the introduction of awareness in groupware systems, each user is equipped with mechanisms, with which he can be aware about activities of other users belonging to a common activity or a common data set. For example, if a user is modifying a shared document, a predefined number of users will be informed about all subsequent changes. For

receiving notification events about status changes, a user first has to subscribe to a notification list. This kind of awareness is also called task-oriented awareness. Awareness can be regarded as a foundation for conflict recognition and resolution. In the field of networked co-operations, where one has a high potential of conflicts and problems among the members, it appears reasonable to have such conflict prevention techniques.

For the CoBE project we have developed the CoBE Awareness Framework (Alda et al., 2004), which realizes a decentral awareness model to coordinate the working activities within a networked co-operation. In contrast to other existing implementations of the awareness model, our system does not rely on a global server that takes over the notification of users with awareness events. Here, the peer-to-peer ideology has also been adopted for our awareness framework: each peer is not only capable of acting as a publisher of awareness events, but also as subscriber who receives events from other peers through single (peer-to-peer) *awareness channels*.

The current version of the CoBE awareness framework can be used by any component-based application that is deployed in the DeEvolve environment (figure 2). The purpose of this framework is actually to make the local interactions between components explicit for other remote users. Users obtaining information about occurred interactions are thus accomplished to derive further actions based on the sole awareness of precedent activities stemming from other users who, for instance, belong to the same co-operation. Users can either be notified directly on the screen or, given that they are unavailable, asynchronously via email. In order to avoid any violation of privacy issues of individual co-operation members, each member is capable of defining so-called *filter agents*. The purpose of these agents is to pre-select events of activities that should not be made explicit for other users. Also, agents allow the selection of non-relevant events stemming from other members.

Apart of the strengths of the awareness model for coordinating planning activities in networked co-operations, a couple of drawbacks have to be faced. First, an awareness model does not assume a global coordination or process model that clearly prescribes when planning activities should be started or finished. Also, no global statements can be made, which of the participating co-operation members should actually be connected through the awareness framework. We simply assume that each participant is responsible on his own to identify and, in turn, to connect to depending members, who work on common data sets or on common activities. As we explain later, these weaknesses can be eliminated in combination with the concepts incorporated in the Bochumer project.

In the Bochumer project the coordination of collaborative design processes is handled by the *workflow agent*. In principle the delegation and coordination of design tasks is accomplished as follows: The workflow agent receives a process template for a given sequence of activities either from the project agent or the project manager. Afterwards, the process template is instantiated with given boundary values. Then, the workflow agent delegates "fireable" design activities (work items) to subscribed design experts subject to the current state of the workflow until the complete workflow is finished.

The applied process support model, thereby, has to fulfill several major requirements that derive from the specific characteristics of collaborative structural design. On the one hand the process model has to support concurrency; on the other hand it has to facilitate time dependent activities e.g. a given task has to be completed at a given point in time. Moreover, the process model has to provide an opportunity to integrate any kind of resource (like product model data, engineering software e.g. for the (semi-)automatic execution of processes, designers, etc.).

We analyzed different Petri net (PN) types with respect to the proposed requirements. As a result high-level, colored, time dependent Petri nets turned out to fulfill our pre-conditions in a convenient way. In particular PN are well suited for systems in which communication, synchronisation and resource sharing are of great importance like in the structural design domain. Another advantage of colored, timed Petri nets is that they can be described with the standardized Petri net markup language (PNML) and provide means for an intuitive graphic representation. Furthermore, it is possible to concatenate several process chains by using the more complex hierarchical colored Petri nets.

Basically, a Petri net is a graphical and mathematical modeling tool for discrete, distributed systems. The structure of a PN consists of places (the passive part of a PN, representing system states), transitions (the active part of a PN, representing events, activities, etc.), and directed arcs. Arcs connect a place to a transition and vice versa. Mathematically, the structure of a low-level PN is represented by a quadruple $PN = (P, T, A, M_0)$ where $P = \{p1,...,p_n\}$ is a finite set of places, $T = \{t1,...,t_m\}$ is a finite set of transitions, A is a finite set of arcs and M_0 is a finite set of initial markings $M_0 = \{m_0^1,..., m_0^n\}$. *Markings* (or sometimes called *tokens*) are attached to places. Tokens are responsible for the execution and, thus, dynamics of the system. The current state of the modeled system (the marking) is given by the number of tokens in each place. If the system state changes at least one transitions "fires". Transitions are only allowed to fire if they are enabled, which means that all the preconditions for the activity must be fulfilled (enough tokens must be

available in the input places). When a transition fires, it removes tokens from its input places and adds some at all of its output places.

The major enhancement of colored Petri nets compared to low-level Petri nets is that they are characterized by distinguishable, high-level tokens of different data types, i.e. places are marked by structured tokens where information is attached to them. More complex high-level Petri nets add individuals, their changing properties and relations to the ordinary (low-level) Petri nets. In other words, tokens have an identity and arcs are labeled with variables, and the transitions may be annotated with a formula. The formula limits the conditions when the transition can fire.

The implemented process support model incorporated in the workflow agent consists of basic high-level Petri net features. For example, it currently supports three types of resources that can be attached to workflow transitions within the *resource manager*: i) subscribed individuals and groups/ organizations, ii) timed resources and iii) an interface for arbitrary application based resources e.g. for the automatic shipping of emails. Besides, arcs can be labeled with formulas that make use of global system variables designed and controlled within the *attribute manager*. The activity manager provides the possibility to link any transition to an implemented, application dependent activity with customized, advanced features. For this purpose, each fireable transition can be mapped to any customized activity class that implements the *activity interface.*

Many design activities in structural design require the preparation of (at least partial) product model data that in many cases cannot be integrated into the Petri net based process model directly. For example, in an instantiated workflow different design activities (fireable transitions/ work items) are attached to the common product model objects that are controlled by the product model agent. Let's assume three of these transitions fire simultaneously and their attached work items are delegated to three different design experts (designer A, B, C) subsequently. Then, in the near future the designers A, B and C have to operate on the same product model objects concurrently without knowing from each other's similar work items. One of the designers (designer A) may start to manipulate some product model objects in co-operation with the product model agent. Let's suppose he finishes its activities before the other designers (designer B and C) start their intended operations. In this case the already permitted operations of individual A can totally differ from the intended operations that designer B and/or C are willing to perform (Fig. 2). This use case may result in severe planning conflicts and inconsistencies.

The above illustrated scenario can easily be improved by deploying a simple notification and awareness mechanism. Every time the designers A, B or C operate on their assigned product model objects the other two designers have to be informed about fundamental manipulations immediately. Such a mechanism can solve or at least reduce planning conflicts and prevent design deficiencies effectively. The above depicted CoBE awareness framework provides exactly such a pluggable mechanism. However, with CoBE subscribed users can perceive all event notifications within a group, but it still depends on them, how to interpret them. In chapter 3 we illustrate how the CoBE awareness framework is integrated into the workflow model and how both approaches are conceptually and technically combined with each other in a favorable and profitable way.

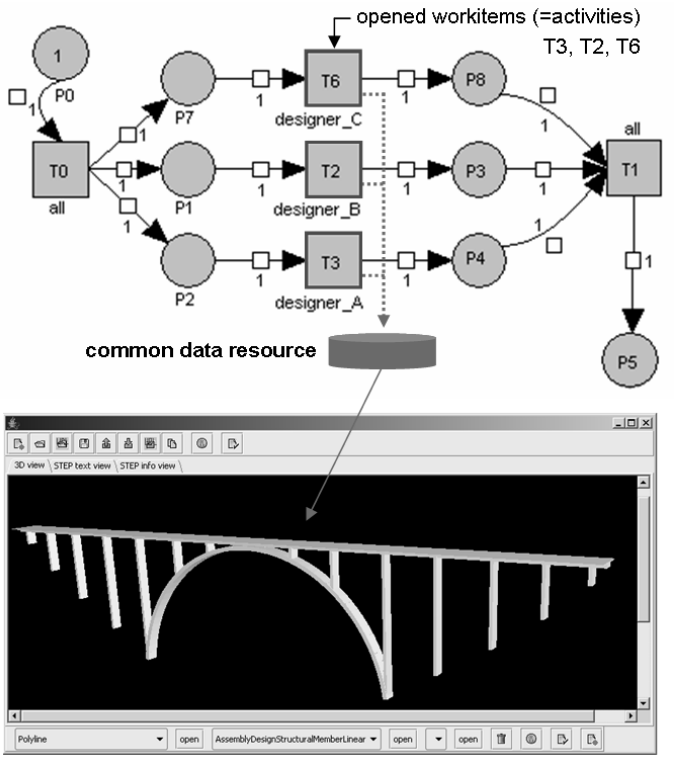


Figure 3: Conflicting use case: designer A, B and C have to operate on the same bridge model objects concurrently.

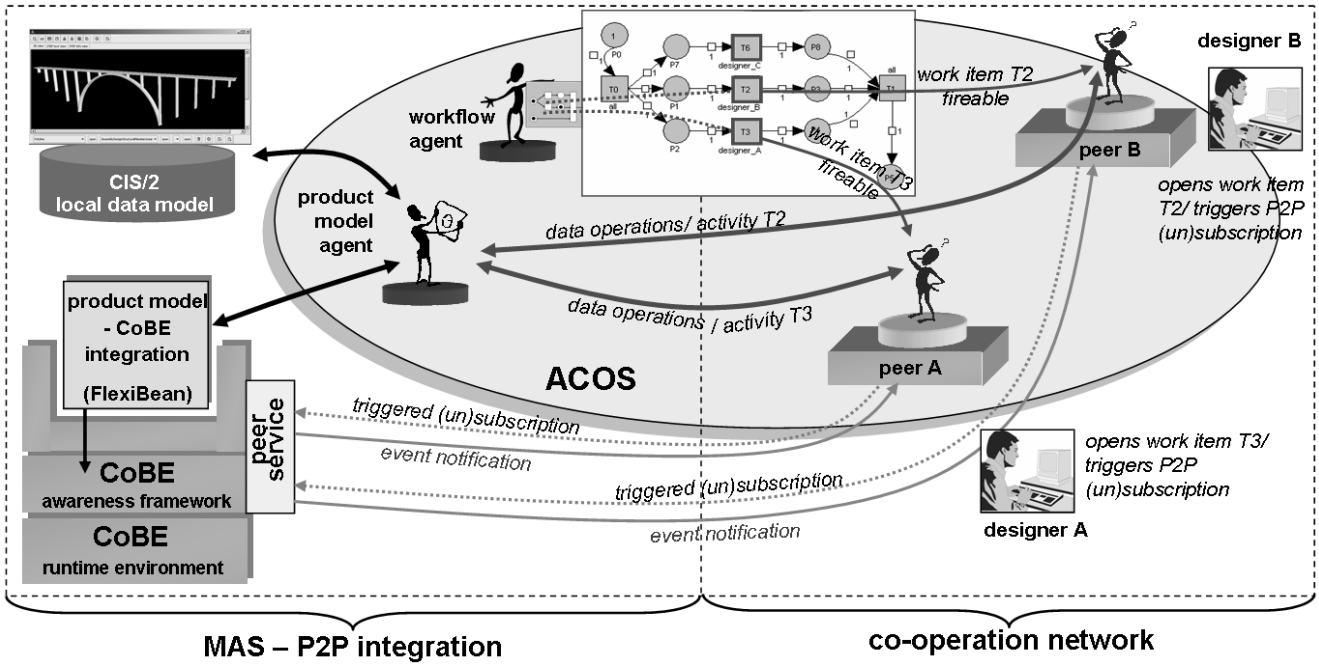


Figure 4. Overall design of the integrated MAS and P2P architecture incorporating enhanced coordination mechanisms.

3 MULTIAGENT AND PEER-TO-PEER BASED WORKFLOW-CONTROL

3.1 Preliminary Work

In the course of our common research activities, we have already proposed and developed an integrated Multiagent and peer-to-peer software architecture *(MAS-P2P)* for supporting collaborative structural design processes within networked co-operations. According to this integrated platform, not only human experts, but also software agents are capable of perceiving awareness events that result from past planning activities. This approach decreases the likelihood that inconsistencies in partial models remain undetected by human actors. Undetected inconsistencies are typically caused by two reasons: (1) human actors are swamped to tackle with too much information and (2) human actors still possess the freedom how and when to react on events. The advantage of agents being able to receive awareness events is motivated by their semi-autonomous characteristic: agents are able to react *immediately* and *autonomously* on events and, accordingly, to react directly on occurred inconsistencies in partial models. More information about the basic assumptions of this platform can be found in (Alda et al, 2004).

3.2 Design Issues

Figure 3 depicts the overall design of the integrated architecture (MAS-P2P) featuring fundamental aspects of both the peer-to-peer and the agent architecture model, respectively. Beyond, the figure points out the incorporated enhanced Petri net based workflow control mechanism and the CoBE awareness model. What one can figure out from the first view is that each participating engineer (e.g. designer A and B) is equipped with an integrated MAS-P2P environment comprising an ACOS personal software agent and a CoBE peer-to-peer runtime environment.

Within ACOS personal agents constitute a kind of gateway to other agents and services like the workflow agent. The workflow agent controls initialized workflows and notifies workflow subscribers about changes in the current workflow state e.g. if an activity was completed and tokens are moved. In the structural design domain a subscriber is primarily an individual personal agent attached to a participating engineer. With that, the workflow agent is able to allocate fireable work items to authorized individual personal agents. The individual structural designers then may decide whether to *open* the received work item or not. Once one of the authorized designers has opened the eyed work item it is no longer fireable. An already opened work item we call *activity*. An activity may be cancelled or closed by its associated design expert. If closed the workflow switches to the next state, if cancelled the work item's state is reset to fireable again.

In structural engineering many activities deal with product model operations. For that reason we implemented the *PMConnectActivity* class, a customized activity that automatically connects a personal agent to the responsible product model agent once a product model based work item is opened. As already indicated it may arise that two or more designers get connected to the same product model agent simultaneously and subsequently are operating on the same common local data model concurrently. In this case it is vital that each designer gets immediately notified when product model objects relevant to its design task have been manipulated by other participating design experts. Additionally, a designer

may only be interested in particular notifications e.g. modifications of particular product model objects or parts of the overall product model. In this case a sophisticated, adjustable filter mechanism is required that accepts only relevant notification events.

For that reason each personal agent is equipped with an individual peer-to-peer runtime environment that easily allows perceiving modeling activities on a common, local model by means of the CoBE awareness framework. Local data manipulation activities, thus, can be made explicit to all (subscribed) members within the P2P co-operation network. For that purpose, all data operations have to be published by peer services. In our MAS-P2P environment the product model agent has to publish all incoming data manipulating actions as a peer service to which other peers – the participating personal agents – may subscribe or unsubscribe.

Technically, the product model agent delegates information about the performed product model operations to a customized *FlexiBean component* (product model – CoBE integration FlexiBean), that is plugged into the CoBE awareness framework. The FlexiBean then is able to trigger appropriate actions like the immediate notification of subscribed co-operation network members every time a data manipulation has been noticed. The following events can be perceived: adding, deleting or modifying of product model objects and state changes of product model object like *locked, unlocked* or finally *released* as well as events indicating that a member has become online or offline. Accessorily, the CoBE awareness framework provides means to filter received information. For instance, a design expert may only be interested in operations on the product model objects with ID #1220 to #1250. Then, the awareness framework automatically filters out events associated to the denoted objects.

However, before the participating designers can receive triggered events they first have to apply for membership to the product model agent's published peer. Obviously, the individual engineers cannot exactly know when to subscribe or unsubscribe to the product model agent's peer in the course of the executing workflow. Instead we enhanced the *PMConnectActivity* with an automatic subscription/ unsubscription mechanism. Every time a participating designer opens a PMConnectActivity work item not only a FIPA-ACL based connection to the relevant product model agent is established automatically but also the inherent designer's peer is triggered to subscribe to the to the product model agent's peer. On the other hand if the considered designer cancels or closes its PMConnectActivity the designer's peer is triggered to unsubscribe from the P2P network.

Let's assume only peer A is subscribed to the P2P network. Then designer A may only be notified about its own product model operations, respectively. Of course, this makes no sense such that these events are suppressed. But assuming designer A and B are both preparing a PMConnectActivity concurrently then designer A gets informed about all the data manipulation actions performed by designer B and vice versa. As we can see the P2P based co-operation network consisting of subscribed peer members dynamically changes due to the current state of the Petri net based workflow. Furthermore, the co-operation network is interconnected automatically without explicit user interactions. It has proved that this mechanism significantly helps to avoid or at least reduce intricatenesses evoked by concurrent, product model based design activities. Thereby, the MAS-P2P environment can only identify potential planning conflicts that finally have to be handled by the affected design experts.

4 CONCLUSIONS

In this paper, we have presented our approach of an integrated architecture model for the effective and efficient support of concurrent activities in the structural design domain. Our integrated MAS-P2P based environment covers the benefits of two major models for coordinating distributed planning activities, that is, an agent-based workflow model and an awareness model.

5 REFERENCES

Alda, S., Bilek, J., Hartmann, D. & Cremers, A.B. 2004, Support of Collaborative Structural Design Processes through the Integration of Peer-to-Peer and Multiagent Architectures, In: Xth International Conference on Computing in Civil and Building Engineering (ICCCBE) 2004, Weimar

Alda, S. & Cremers, A. B. 2004, Strategies for Component-based Self-Adaptability Model in Peer-to-Peer Architectures, In: Proceedings of the 4th International Symposium on Component-based Software Engineering (CBSE7). Springer. Edinburgh, Scotland

Bilek, J. & Hartmann, D. 2003, Development of an Agent-based Workbench supporting Collaborative Structural Design, In: The 20th CIB W78 Conference on Information Technology in Construction, New Zealand, Waiheke Island

Dourish, P. and Bellotti, V. 1992, Awareness and Coordination in Shared Workspaces, In: Proceedings of the Proceedings of the 4th ACM Conference on CSCW. Toronto / Canada

FIPA 2000, Foundation for Intelligent Physical Agents (FIPA) 2002 standards, http://www.fipa.org

Reed, K.A. 2002, Role of the CIMsteel Integration Standards in Automating the Erection and Surveying of Constructional Steelwork, In: Proceedings of the 19th International Symposium on Automation and Robotics in Construction (ISARC), National Institute of Standards and Technology, Gaithersburg, Maryland, p. 15-20

Sun 2004, JXTA v2.0 Protocols Specification. (http://spec.jxta.org/v2.0/)

Van der Aalst, W.M.P. 1998: The Application of Petri-Nets to Workflow Management, In: Journal of Circuits, Systems and Computers 8 (1998), Nr. 1, p. 21–66