THE DEEVOVLE PLATFORM: ENSURING THE CONSISTENCY BETWEEN DISPERSED PLANNING ACTIVITIES IN STRUCTURAL DESIGN

Sascha Alda[1], and Armin B. Cremers[2]

ABSTRACT

This paper aims at presenting the DEEVOLVE platform. DEEVOLVE is a service-oriented peer-to-peer architecture that allows for deploying standard CAD software in terms of a peer service into a networked co-operation. This platform accomplishes the collaborative sharing of design models and the reasoning about design decisions in projects, which do not possess a central authority. Owing to the strengths of DEEVOLVE to handle anticipatable exceptions potentially occurring within a networked co-operation, the platform is suitable especially for virtual project structures exhibiting fluctuating constellations and a high degree of heterogeneity. Besides summarizing recent developments and concepts of DEEVOLVE, this paper mainly focus on presenting two dedicated scenarios derived from a general use case originating from a real life structural design project. It is shown, how the proposed platform can deal with and, moreover, how it can improve the effectiveness of the presented scenarios.

KEY WORDS

Software architectures, distributed coordination, structural design, conflict management.

INTRODUCTION

Considering the way of organizing up-to-date projects in structural design, one can clearly notice a trend towards project structures encompassing a huge number of dispersed project partners, external experts and other facilities. These so-called *virtual organizations* (VOs) carry out design activities in a collaborative way. The aspired benefits of virtual organizations is to increase the degree of parallel planning activities (carried out by selected experts) leading to a shortening of development time and, thus, a reduction of the overall project costs. The collaboration of the participating partners is commonly enhanced by group-supporting software systems that enable to share CAD documents or reason about important design decisions. One of the distinguished requirements of these software systems is to integrate heterogeneous software installations (e.g. CAD systems, simulation tools) towards a global *homogeneous networked co-operation* in which people can share any kind of working items. To reach this goal, modern software systems nowadays make use of

[1] Research associate, Computer Science, Institut for Applied Computer Science, University of Bonn. Römerstraße 164, 53117 Bonn, Germany, Phone +49 228/ 73-4299, , alda@cs.uni-bonn.de

[2] Professor, Computer Science, Institut for Applied Computer Science, University of Bonn. Römerstraße 164, 53117 Bonn, Germany, Phone +49 228/ 73-4501, FAX +49 228/ 73-4382, abc@cs.uni-bonn.de

standardized notation for resource description (e.g. XML), exchange formats (e.g. IFC) or interaction protocols (e.g. SOAP). Recent representative implementations of such software systems (c.f. ACOS (Bilek and Hartmann, 2003), MADITA (Theiss *et al.*, 2004)) are based on the Java platform that provides sophisticated implementations of those standards.

A major problem that hinders the definite appreciation and success of virtual organizations in the area of construction informatics is the *remaining degree of heterogeneity* and the *dynamic nature* in virtual organizations. Despite the presence of the above-mentioned standards, partners often insist on working with proprietary software, use incompatible document exchange formats, or have various hardware infrastructure. Moreover, partners still possess the freedom to adapt software arbitrarily (e.g. deploying a new version of a CAD software). While such adaptations to a local environment may necessarily improve the efficiency at a single partner's site, it may violate context dependencies to other partners who depend on the original software version. In construction informatics, notions for analyzing and determining such dependencies in order to avoid such context violations are yet missing.

The dynamic nature of virtual organizations reflects the fact that partner may join and leave a co-operation depending on their dedicated role in a virtual organization, due to project-related circumstances, or due to individual reasons (e.g. insolvency of a partner). Besides these rather medium-term reasons, there are also short-term reasons for leaving a co-operation, for instance, evoked by temporary network connections. Owing to the dynamic availability of partners and their resources, effective planning activities often cannot be ensured in such virtual project structures. Existing tracing or monitoring concepts can be applied but only provide for a rudimentary support. What is clearly missing is a formalism to define rules expressing global semantic dependencies between resources and stakeholders that must be met during certain planning activities within a given networked co-operation.

In this paper, the DEEVOLVE platform is presented (Alda and Cremers, 2004)[3]. DEEVOLVE is based on the service-oriented peer-to-peer architectural style. Following this style, each partner of a networked co-operation is equipped with a peer environment enabling him to provide as well as to use services (e.g. interface to access CAD files or verification algorithms). DEEVOLVE features the notion of an *adaptation policy* to regulate the adaptation of services and, thus, to limit the risk of dependency violations. Furthermore, *integrity constraints* are introduced to define global states on a networked co-operation. DEEVOLVE uses these constraints to ensure the consistency among coherent design documents in a networked co-operation. The intention of this paper is to present the current status of the DEEVOLVE platform. As a novel, previously unpublished contribution, two concrete application scenarios are presented to demonstrate the capabilities of DEEVOLVE for supporting networked co-operations from the area of structural design.

The present paper is structured in the following. Section 2 presents the core concepts of the DEEVOLVE platform. Section 3 then elaborates the two scenarios stemming from the area of structural design in order to demonstrate the capabilities of DEEVOLVE. Section 4 finally presents a conclusion of the results presented.

[3] DEEVOLVE is the main contribution of the CoBE project. This project is funded in the course of the priority program 1103 „Networked-based co-operative planning processes in structural engineering" by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG).

THE DEEVOLVE SOFTWARE PLATTFORM

ESSENTIAL PARTS OF DEEVOLVE

DEEVOLVE follows the notion of a service-oriented peer-to-peer architecture. Service-oriented peer-to-peer architectures refer to the class of software architectures that feature a set of equal nodes or so-called *peers*. Conventional end-user personal computers or terminals typically represent such peers. Each peer is capable of functioning as a provider and as consumer of an arbitrary number of *peer services* encapsulating functions, hardware routines, or public access to documents. Consumed services can be composed to new, more complicated applications or even new services that can in turn be located and used by other third-party peers (see typical constellation in figure 1). This architectural style constitutes a logical enhancement of traditional peer-to-peer architectures relying on the provision of a small number of services (Brookshier *et al.*, 2002) with novel concepts from the area of service-oriented architectures (Cervantes and Hall, 2005), in particular service composition and service description. In contrast to centralized service-oriented architectures, peer-to-peer architectures assume an unstable and dynamic topology as an important constraint due to the sole responsibility of peer owners to connect to a network. Since users are directly involved while a peer is operating, they should be considered as active *decision makers* especially during important management activities (e.g. exception handling, explained later on).

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.

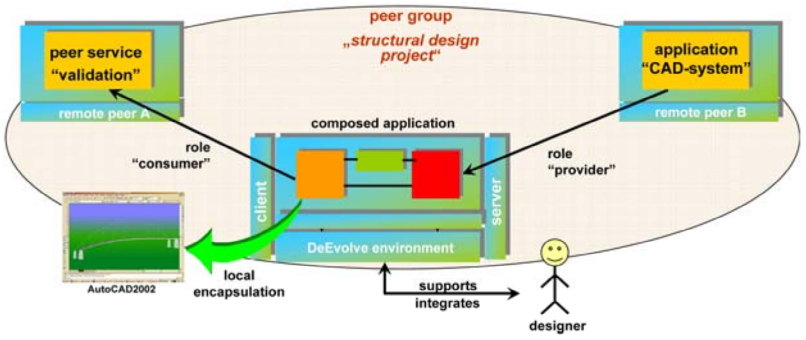


Figure 1: The structure of the DEEVOLVE platofrm: users are able to consume and to provide services.

Conceptually, DEEVOLVE is based on top of the JXTA standard peer-to-peer framework (Sun, 2005). DEEVOLVE incorporates the *component methodology* as another technical fundamental basis. According to the component approach, peer services can be created by the composition of single software components. A component model called FLEXIBEAN prescribes 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 peers are able to discover and use these services. Additionally, we provide the composition language PeerCAT enabling users to declare the composition of different peer services towards high-level applications.

Exception Handling (Integrity Constraint)

As a self-adaptable architecture, DEEVOLVE is capable of detecting and handling exceptions in particular the failure of dependent peers, that is, peers from which a consumer peers is using a provided service. The detection of exception is achieved by a *monitoring mechanism* that traces the availability of dependent provider peers. Any detected exception can be forwarded to peers that may indirectly dependent on the functionality of that lost peer.

The handling of exceptions is done in strong interaction with end-users, that is, the operator of a peer. At selected decision points, users are able to decide which routines need to be executed for handling an occurred exception. For the actual handling procedure, the platform uses the same component-oriented operations as for creating the initial service composition (e.g. discover and add new service, bind two services, change parameters to a service). DEEVOLVE is accompanied by a couple of auxiliary tools that utilize these methods.

Operators of peers are also accomplished to define integrity conditions on service compositions. Integrity conditions define sound (correct) (sub-)compositions that must be fulfilled during use time of a composition. The violation of an integrity condition (e.g. caused by the failure of a peer), denotes an exception. Exception handlers that can be associated to an integrity condition that can handle such an exception. Again, DEEVOLVE is able to forward the event of an integrity violation to all dependent peers. Due to their global scope, (they may potentially comprise many peers), integrity conditions represent a discrete state a peer-to-peer architecture may enter at a single point in time. This concept is practical in such peer-to-peer architecture, where peers act in a collaborative manner to achieve a defined goal. For the time being, we have developed two concrete (default) types of integrity conditions, among others, the “Minimal_Composition” type, which is introduced later.

Dependency Management (Adaptation Policy)

Each operator of a single peer owner is capable of adapting (public) peer services according to changed requirements. Before an adaptation request can actually be proceeded with, an *analysis of existing consumer dependencies* has to be performed. It aims at suppressing the violation of functional dependencies resulting from dependent consumer peers. The analysis is based on consumer information that has been passed during the process of subscribing (binding) from a consumer to a provider of a peer service. Consumer information particularly includes a *weighted rating* indicating the relevance to a consumed (dependent) service.

The proposed model furthermore assumes that peers belonging to a self-organized peer group have agreed on a global *adaptation policy* in which conditions are formulated that entail procedures on how to cope with existing (weighted) dependencies (Alda, 2005a). Procedures incorporate, for instance, the notification of dependent consumers before the adaptation is processed. Alternatively, the stakeholders of a peer group can negotiate about the planed scale or the rationale of the inquired adaptation request. For both the analysis of dependencies and the subsequent adaptation, appropriate tools have been developed.

If the adaptation of a service is performed in spite of the implied demands and procedures of an adaptation policy, then the policy is violated. In this case, all dependent and affected consumers are able to argue a negative *reputation value* against this provider. This reputation is spread as an advertisement to all rendezvous peers (i.e. founders of peer groups). Any peer can request the reputation of other peers. Based on the gained reputation, a peer is able to decide whether to trust to a peer (operator) and assess its quality of service.

Peer Services for Structural Design

In order to integrate existing legacy applications, we deploy so-called bridge components. Such a component allows for mediating between the standardized FLEXIBEANS interface and other interfaces such as Microsoft's COM interface. To this end, we are able to encapsulate widely used CAD applications like AutoCAD 2002 in a peer environment and to publish it as a peer service into the peer-to-peer architecture (see figure 1). This feature allows for directly sharing CAD-based documents among peers (peer service "*DocExchange*"). Also, it serves as the basis for assimilating information about the progress of individual modeling activities to dependent peers (peer service "*AwarenessData*"). The latter peer service uses the COBE AWARENESS FRAMEWORK in order to assimilate awareness data to subscribed user. This service (and the framework) serves as the foundation for coordinating the activities of partners within a networked co-operation (see (Alda, 2005b) for more information).

APPLICATION SCENARIOS

This section illustrates two concrete scenarios stemming from the area of structural design in order to demonstrate the capabilities of DEEVOLVE'S concepts shown in the previous section. In general, an *(application) scenario* is a concrete, focused and informal description of how a system might be used (Bruegge and Dutoit, 2004). A *use case* is an abstraction of many coherent scenarios describing the overall functionality of a software system. The technique of describing the functionality of a software system by scenarios and use cases is utilized during requirements engineering for eliciting and describing the requirements of a system.

In the context of this paper, scenarios and use cases are taken to visualize suitable planning scenarios in which planning activities to a set of coherent partial design models are carried out concurrently. Later on, it is shown, how DEEVOLVE can support these scenarios. At an early stage of the CoBE project, these scenarios have served as a foundation for understanding the application domain of "structural design" more accurately and to concretize the design and the implementation of the DEEVOLVE platform. We first outline a general use case that encompasses the basic set up of a suitable networked co-operation. Then, two concrete scenarios are explained focusing on a useful application of both the adaptation policy and the integrity constraint concept, respectively.

USE CASE "GENERAL SET-UP OF NETWORKED CO-OPERATION"

"*Three different structural designers A, B, C and a project leader D (an architect) are willing to collaborate within a networked co-operation. Their aspired goal is to produce a coherent structural model of a steel bridge construction (more accurate, technical data of this bridge can be read in (Alda, 2005b)). The steel bridge necessitates the involvement of further yet unknown specialists (like expert X) for verifying complex abutment elements of* *that bridge. Here, deficient partial model are expected that could lead to subsequent modifications on all partial models. Any additional expert should not stay in the group permanently. Three partners agree to work with AutoCAD 2002, partner A still works with AutoCAD 2000. All partners exchange their partial models w.r.t. IFC 2.2. All partners expect to be notified about changes on partial models. All partners tend to work synchronously but at different locations*."

Realization in CoBE and DEEVOLVE

The general structure of a distributed setting of the DEEVOLVE platform supporting this use case is depicted in figure**Fehler! Verweisquelle konnte nicht gefunden werden.**.

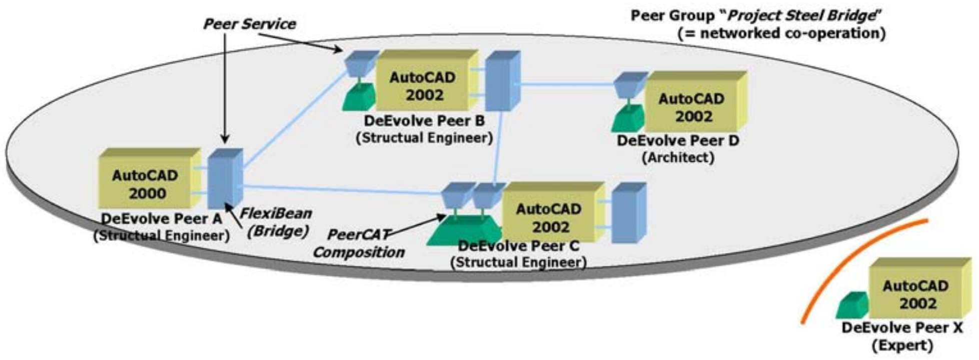

Peer Service
Peer Group "Project Steel Bridge" (= networked co-operation)
AutoCAD 2002
DeEvolve Peer B (Structual Engineer)
AutoCAD 2002
DeEvolve Peer D (Architect)
AutoCAD 2000
DeEvolve Peer A (Structual Engineer)
FlexiBean (Bridge)
PeerCAT Composition
AutoCAD 2002
DeEvolve Peer C (Structual Engineer)
AutoCAD 2002
DeEvolve Peer X (Expert)

Figure 2: The structure of DEEVOLVE for supporting the general use of a networked co-operation

Each partner A - D within the co-operation is equipped with a DEEVOLVE peer environment that is to be installed on each terminal. The same terminal deploys the local AutoCAD 200X installation. The JavaToCOM bridge of DEEVOLVE allows to integrate AutoCAD in the peer environment. AutoCAD can now be published as a peer service to all other participants of the co-operations. From the given set of default peer service, for instance, the "*DocExchange*" service could be used to share AutoCAD-based documents (IFC) among the peers. A peer group represents this co-operation that defines the boundaries of it. This way, advertisements of peer services are only published within that peer group. The architect (peer D) serves as the founder of that group. He authorizes new partners when they are willing to join the group. Only members being authorized are able to access and to bind a peer service in their environment. Thus, expert X first has to apply for membership before he can join the group.

APPLICATION SCENARIO 1 "HANDLING DYNAMIC AVAILABILITY OF PARTNERS"

"*The modeling of the northbound supporting node attaching the steel deck construction to the deck abutment involves two different partners (A, B) and an external expert (X) (see figure 3 below). Partner A details the structural steel elements of the deck. Partner B designs the concrete abutment below the deck in this section. Certain subtle modifications of the steel deck (e.g. varying the height or width of the steel deck) could effect the reinforcement detailing of the underlying concrete abutment. On the other hand, new evaluation data concerning the soil condition of the ground (acquired by partner A) could not only influence the fundament of the abutment but also effect the overlying steel deck (e.g. its maximum weight). An external structural design expert (expert X) must assess each anxious* *modification on either model before subsequent modifications on dependent models are performed. Owing to the pressure of the project, both assessment and execution of subsequent modifications should be handled synchronously without any serious delay. Architect and project leader D would like to be notified on potential changes and also on critical delays."*

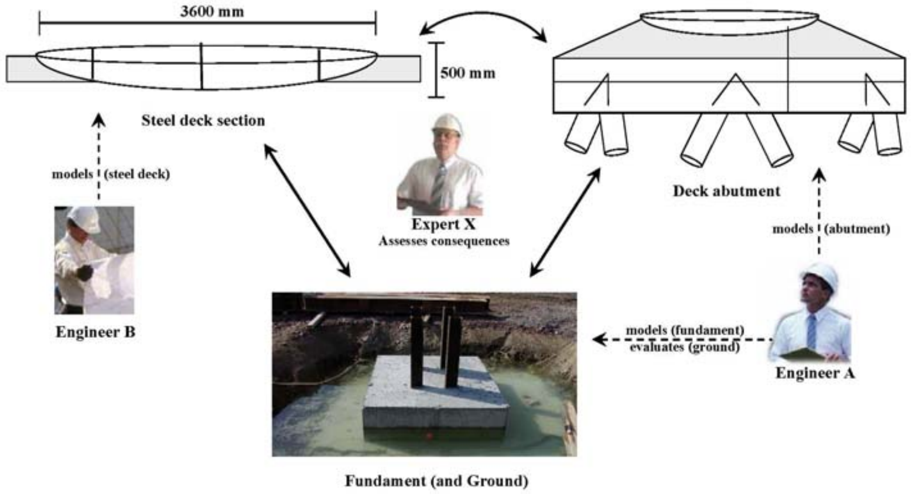


Figure 3: Visualization of scenario 1: involvement of three engineers during the concurrent planning of a supporting node. Architect D as the project leader is not depicted.

Realization in DEEVOLVE

For supporting this subgroup of planers and expert(s), each of those stakeholders must be connected with each other to exchange partial design models and status information. Technically, this can be achieved by locating and binding the respective peer service "DocExchange" or "AwarenessData" provided by each partner and expert into the local peer environments. The corresponding PeerCAT file from partner A is depicted in figure 4:

```
<?xml version="1.0"?>
<!DOCTYPE composition PUBLIC "" "http://cs.uni-bonn.de/~palij/deevolve/composition.dtd">
<composition name="Local-DocExchange" id="newUUID">
 <service name="DocExchange" type="flexibean" id="Expert_X">
  <accessPorts>
    <port componentID="proxy" name="getDoc" type="DocListener" polarity="required" protocol="event"/>
    <port componentID="proxy" name="pushDoc" type="DocPusher" polarity="provided" protocol="event"/>
  </accessPorts>
 </service>
....
 <rbind id="1">
   <port1 componentID="proxy" name="getDoc" serviceID=" Expert_X "/>
   <port2 componentID="client" name="getDoc" serviceID="Partner_A"/>
 </rbind> …
```

Figure 4: PeerCAT sniplet defining the composition between Partner A and Expert X

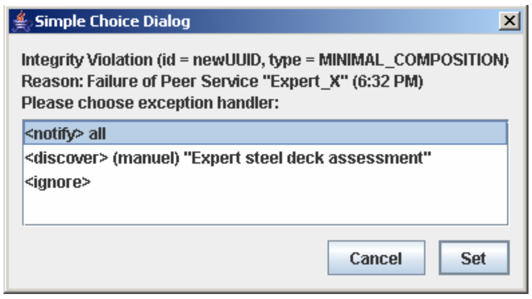
In this PeerCAT file, the "*DocExchange"* peer service of expert X is bound with the local *DocExchange* application of partner A, so that models between these two applications can be exchanged. The port *getDoc* can be used by the expert X in order to request the current design model of partner A. Alternatively, partner A can actively push the design model to the expert's environment by invoking port *pushDoc* (concrete binding is omitted).

```
<composition name="Local-DocExchange" id="newUUID">
…
 <semantics>
  <integrity type="MINIMAL-COMPOSITION" id=newUUID" class="minimalComp.class">
    <serviceID>Partner_B</serviceID>
    <serviceID>Partner_D</serviceID>
    <serviceID>Expert_X</serviceID>
  </integrity>
 </semantics>…
```

Figure 5: Declarative description of the integrity condition describing a minimal composition

Declarative exception handlers can enrich the binding to the peer service of expert X. After deploying the composition, the peer of expert X is monitored. Each time a failure of that peer is recognized, the respective handlers are invoked. The assumed concurrent modeling of the supporting bridge, however, implies the co-operation of additional stakeholders namely partner B and the architect D. For all those partners, appropriate declarative bindings have to be defined so that design models and information can be exchanged among these stakeholders. This clique indeed represents a *minimal composition* that must be fulfilled, especially during synchronous modeling. In order to formulate and to be aware about failures within this clique, an integrity condition of type "Minimal_Composition" can be used in accordance to the local composition. For partner A, the integrity condition is to be phrased as illustrated in figure 5. The condition is met, if the links to all peer services summarized in the semantics tag are available. The set of services defining the minimal composition must be a subset of services that is defined in the corresponding PeerCAT composition. The condition is violated if at least one peer service is unavailable. In this case, an exception is thrown.

If the integrity condition is fulfilled, then all pertaining stakeholders are able to exchange data and information from their modeling activities in a reliable manner. This way, the consistency among all partial building models (here: models modeling the abutment and steel desk) can be ensured. Also, additional value-added service can be bound and used steadily (here: the assessment of dependencies between the steel deck and the abutment by expert X).



Integrity Violation (id = newUUID, type = MINIMAL_COMPOSITION)
Reason: Failure of Peer Service "Expert_X" (6:32 PM)
Please choose exception handler:

<notify> all
<discover> (manuel) "Expert steel deck assessment"
<ignore>

Figure 6: Dialog from which a partner can choose the most appropriate exception handler.

If the integrity condition is violated, then each partner can choose among a fixed number of exception handlers. Both exception handlers and the integrity condition itself can be predefined by architect D and can be made available to all stakeholders. The exception handlers are also defined the PeerCAT file (omitted here). During deployment, DEEVOLVE interprets and transforms these into executable Java code. Given the violation of an integrity condition, a dialog is presented to the partner who has detected the violation (figure 6).

In the dialog of figure 6, the *id* and the *type* of the violated integrity condition are displayed as well as an indication of the causing reason (here: the failure of peer service provided by expert X). Based on this information, the partner can among three handlers:

- <notify> all allows a user to announce and thus to forward the exception to all partner being involved in the minimal composition (the user can select between Email or on-screen notification in the subsequent dialog). This way, all other partners also become aware about violations. Based on this information, they can make their individual conclusion on how to cope with this exception.
- The <discover> (manual) "Search keys" handler allows for locating an alternative peer service that fulfills the indicated search keys. This option becomes relevant in situations, when the depending assessment service from expert X has become repeatedly unavailable and, thus, turns out to be unreliable. Especially during a tight project schedule, it could be beneficial to discover an alternative expert that offers an alternative peer service to evaluate modification on steel constructions. The search for new services is initiated by DEEVOLVE, the results are displayed accordingly. The manual attribute dictates that the user is then able to select the most suitable service and to bind it in the local environment.
- The <ignore> handler allows for ignoring the exception. This can be applied, if the violation is actually not relevant at a given point of time.

Additional Support: The COBE AWARENESS FRAMEWORK

Besides the ability to control the availability of partners and to define integrity conditions on co-operations, a peer group may fall back on the COBE AWARENESS FRAMEWORK to perceive concrete modeling activities of dependent partners. To subscribe into the awareness channel of a partner, a consumer is asked to locate and to use the peer service "*AwarenessData*" of the respective partner and to register as an interested user. Then, the user will be notified about important changes on design documents, saving and opening of documents and so on. Again, the capability of DEEVOLVE to monitor the peer group's availability (state) is also profitably for the awareness framework. The synchronous collaboration of partners by means of the awareness framework can only be established if the peers are reachable. The concrete possibilities of the awareness framework can be looked up elsewhere (e.g. (Alda, 2005b)).

APPLICATION SCENARIO 2 "HANDLING ADAPTATION REQUESTS"

"*During the last third of the project, partner A decides to upgrade his local AutoCAD 2000 installation to version 2003. Obviously, this upgrade promises an improvement of his working environment. Partner A would like to ensure that this upgrade does not violate any contextual dependencies within the given co-operation (peer group).*"

Realization in COBE and DEEVOLVE

Indeed, the update as described in the second scenario leads to various problems. The COBE AWARENESS FRAMEWORK makes use of the COM-based event channels of AutoCAD to determine incidents on modifying documents and to fetch a copy of the current design model. In newer AutoCAD versions (from 2003), however, the COM interface has been removed. So, composing newer AutoCAD versions with our awareness framework is no longer feasible. Consequently, partners of the peer group could no longer be notified about modifications on design models. Besides this software compatibility concern, the leader of that peer group (architect) may be worried about the compatibility of the exchange formats. Although all AutoCAD versions apply the IFC standard, incompatibilities may occur due to improper implementations especially of rarely used aspects. The availability of incompatible documents is a problem when merging the partial models to a complete model.

The DEEVOLVE platform utilizes the notion of adaptation policy to avoid potential risks when adapting local software. In this scenario, the architect acting as the group founder is responsible to define and to provide the adaptation policy for his group. The selected adaptation strategy clearly depends on the current project status. Towards the end of a (critical) project, the architect could update the adaptation policy imposing a more restrictive dealing with adaptation requests. A possible strategy could entail that each adaptation request should be negotiated with all partners of the group. An extension to that policy could state that, given at least one highly weighted dependency to a service, no adaptation is allowed.

CONCLUSIONS

This paper has presented the latest concepts of the DEEVOLVE platform. We could show, how DEEVOLVE can actually improve the effectiveness of dispersed planning activities by means of a real life structural design projects. Compared to other system that promise to support virtual project structures, DEEVOLVE aims at limiting possible consequences due to the novel concepts of exception handling, integrity constraints, and adaptation policy.

REFERENCES

Alda, S. (2005a). "Peer Group-Based Dependency Management in Service-Oriented Peer-to-Peer Architectures", in: Proc. of the *Third Int'l Workshop On Databases, Information Systems and Peer-to-Peer Computing (DBISP2P), in conjct. with 31st Int'l Conf. on Very Large Data Bases.* Trondheim, Norway.

Alda, S. and 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 (LNCS 3054). Edinburgh, Scotland. (pp. 59-67).

Alda, S. C., A.B., Bilek, J., and Hartmann, D. (2005b). "Integrated Multiagent and Peer-to-Peer based Workflow-Control of Dynamic Networked Co-operations in Structural Design", Proceedings of the *22nd Int'l Conference Information Technology in Construction (CIB-W78).* Dresden, Germany.

Bilek, J. and Hartmann, D. (2003). "Development of an agent-based workbench supporting collaborative structural design", Proceedings of the *20th CIB W78 Conference on Information Technology in Construction*. Auckland, New Zealand.

Brookshier, D., Govoni, D. and Krishnam, N. (2002). "JXTA: Java P2P Programming", in: (eds.): SAMS.

Bruegge, B. and Dutoit, A. H. (2004). *Object-Oriented Software Engineering: Using UML, Patterns, and Java.* Pearson Prentice Hall, London.

Cervantes, H. and Hall, R. S. (2005). "Technical Concepts of Service Orientation", in: Stojanovic, Z. and Dahanayake, A. (eds.): *Service-Oriented Software System Engineering: Challenges and Practices.* IDEA.

Sun (2005). *JXTA v2.0 Protocols Specification*. 2005. (http://spec.jxta.org/v2.0/)

Theiss, M., Meissner, U. and Rüppel, U. (2005). "Network-Based Fire Engineering Supported by Agents", Proc. of the *10th Int'l Conference on Computing in Civil and Building Engineering*. Weimar, Germany.