Mobility of Context for Project Teams

Schahram Dustdar
Caramba Labs Software AG
Seidlgasse 21/25, 1030 Wien, Austria
dustdar@CarambaLabs.com

Abstract

In the last decade, bureaucratic organizational hierarchies increasingly have been replaced with flatter organizational forms, bringing together people from different disciplines to form project teams within and between organizations. Distributed project teams often are self-configuring networks of mobile and "fixed" people, devices, and applications. They are the natural next step in the evolution of distributed computing, after client-server, Web-based, and peer-to-peer computing. A newly emerging requirement is to facilitate not just mobility of content (i.e. to support a multitude of devices and connectivity modes) to project members, but also mobility of context (i.e. to provide traceable and continuous support of relationships between people, artifacts, and business processes). The contribution of this paper is to present the design goals, the architecture, and implementation of a system aiming at supporting mobility of context for project teams, enabling traceable and continuous support of associations (relationships) between people, artifacts, and business processes.

Keywords: Project teams, mobility of context, Workflow, Groupware

1. Introduction

In the last decade, bureaucratic organizational hierarchies increasingly have been replaced with flatter organizational forms, bringing together people from different disciplines to form project teams within organizations. These teams are under heavy pressure to increase time-to-market of their products or services and lower their coordination costs. In addition to new organizational forms within corporations, increasing globalization of the economy and new Internet technologies lead to groups with members scattered across organizational boundaries and in different time-zones [5,14]. This requires efficient coordination among members and systems, which needs support and integration of (legacy) applications, application servers, directory services, and corporate databases.

Distributed project teams often are self-configuring networks of mobile and "fixed" people, devices, and applications. They are the natural next step in the evolution of distributed computing, after client-server, Web-based, and peer-to-peer computing. A newly emerging requirement is to facilitate not just mobility of content (i.e. to support a multitude of devices and connectivity modes) to members, but also mobility of context to distributed project teams. With mobility of context we mean a *traceable and continuous* support of associations (relationships) between people, artifacts, and business processes. Context is composed of information on the "who, when, how, and why". In order to illustrate the lack of context, consider an "Explorer"-like view on a file system. This view allows the person to see documents (artifacts) stored inside folders. The name of such folders might reflect project names themselves. The mentioned view on these documents does not contain further contextual information on what a person (yourself, or others) actually have to do (did) with it (e.g. create another document, send an e-mail to customer, call partner organization, etc.). For example if the person in the above example needs to see who actually received a document stored in any given (project) folder, he is required to manually retrieve his e-mail box in order to find this information. This simple example shows that relationships (links) between artifacts, such as documents or database information, and activities performed by persons are usually not stored in groupware, project management or workflow management systems. However this linkage is of paramount importance for knowledge-intense business processes of project teams in order to provide contextual information on knowledge artifacts for processes such as new product development, which cannot be modeled using a traditional workflow management system.

The remainder of the paper is organized as follows. The next section briefly discusses related work. In section 3 the design goals and an architectural overview are discussed. Section 4 presents the systems' client-, administration-, and service components. Finally, section 5 presents the conclusions and future work.

2. Related Work

To our knowledge traditional groupware, workflow management systems, or e-mail systems do not support the requirements outlined above. Most groupware systems follow a "workspace" metaphor, which allows users to upload/download artifacts using files and folder to organize their work. When e-mail is used as the main medium for project teams (as in most cases), data and associated information (such as attachments) remain on central mail servers and/or personal inboxes without any *context* information in which those email communications were used (involved business processes, performed activities, created artifacts as described above). Enterprise groupware systems are generally focused on enterprise-wide messaging and discussion databases and do not support organizational components and structures such as people and their associated roles, groups, task, skills etc. This leads to "organizationally unaware" systems treating all messages alike (semantically) and without any awareness of underlying business processes, which are essential for efficient collaboration in project teams. Workflow systems support the notion of processes within an organization [1, 2, 3, 7, 8, 15, 16]. However, they require to first model a business process (build time) and then to enact this model (run time). This leads to incredible inflexibility [9] for project teams. In business "exceptions are the rule", therefore modeling a process is often not possible for creative, innovative project teams of knowledge workers such as in product development or consulting teams.

Capturing the process by which knowledge is collaboratively developed (knowledge creation process) is as important as documenting the output of group collaboration or personal work. In other words, "results" must provide more than data and information - they must also serve as containers for documenting the "how we arrived" at the outcome. Today email is used as the main mechanism for exchanging ideas, concepts, and knowledge work in project teams. However the missing mobility of context increasingly attracts more attention.

The goal of this paper is to provide a brief overview of the design goals and an architectural overview of *Caramba* [4, 10]. Software prototype development began in 1997 and it evolved into a commercial product, which has been launched in 2001. Caramba manages all involved processes in knowledge work for project teams: from creating ideas, via using enterprise applications to support this work, up to coordinating and making this processes visible and reusable both within, and between organizations.

3. Design goals and architecture

The main design goal encompasses the support for mobility of context. Therefore support for meta-modeling the organizational (team) structure as well as the ability to integrate business objects (e.g. DBMS-tables) is essential. Secondly, traceable and continuous support regarding the relationships between people, artifacts, and business processes is of paramount importance. Thirdly, different levels of corporate integration with other information systems (e.g. SMTP-server, Web Server, etc.) should be possible. Finally, the system should allow outside project partners to be integrated with the project team as tightly as possible, allowing access to all information provided by a CarambaSpace, if security policies allow. Various access mechanisms such as using a web-browser, Java client application, or mobile device have to be provided.

In the following section we will provide an overview of architectural issues of *Caramba* [4, 10]. An in-depth presentation of the architecture or the software itself is beyond the scope and focus of this paper. The software (middleware and client) is written in Java based on Java SDK 1.2.2 for enhanced GUIs using the Java Foundation Classes (JFC) and to support drag and drop. Software architectures typically include the description of *components*, *connectors*, and *configurations* [12, 13]. For this it is important to decompose a system into a well-defined set of components that have clear responsibilities [11]. Since architectures for project teams have to integrate with various information systems installed in organizations, we decided to strive for a middleware style rather than a classical client-server style.

Figure 1 provides a high-level overview of supported setup scenarios. The left part shows a simple application scenario with one project team (and therefore one CarambaSpace) using an embedded Java-DBMS. Caramba services are hosted on one server. Clients may access the project's team-space using Java client applications or via a built in HTML/XML portal. In cases where tight integration with corporate information systems and databases is required, the administrator utilizes the Caramba meta-modeler and relation wizard to integrate corporate DBMS and other resources such as SMTP-Mail servers or company Web Servers (see right hand side of Figure 1).

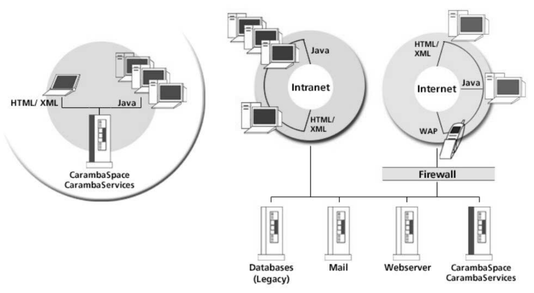


Figure 1. Integration scenarios

The following descriptions will point out the respective architectural style used in a particular layer or component. The Caramba software architecture [10] is composed of multiple layers: middleware, client suite, and a persistence store, as depicted in Figure 2. Objects and services are accessed through the ***Transparent Access Layer*** (TAL) from the CarambaSpace platform (middleware). Depending on access mechanisms and the requested services (e.g. via Java client with RMI protocol or via Web browser with http), Caramba provides a unique way to handle requests using a meta-model framework to describe content and separating presentation, logic, and data. This model permits high flexibility, enables customization, and extensions as well as the adoption of new devices or technologies.

The goal of this layer is to offer transparent access to a CarambaSpace. The TAL utilizes various services to transform, describe, manipulate, and observe objects. All objects managed through a CarambaSpace are well described using a meta-model description framework. Objects can be customized in their structure (e.g. adding columns to tables, adding relations to objects) and their presentation by adopting their meta-model description. Any changes are dynamically reflected by client components. Based on the meta-model description framework, Caramba enables various options to customize data and content as well as to integrate data from different resources (e.g. corporate databases). This layer also provides facilities for fine-grained object notification services and the implementation of customized services based on object observers.

The middleware however, does not manage states and persistence of objects itself. Objects are stored, manipulated, and retrieved via the **Persistence Layer** (PEL). Caramba leverages and adopts standard Java based technologies (e.g. JDBC, JNDI, HTTP, etc.) to access and integrate data.

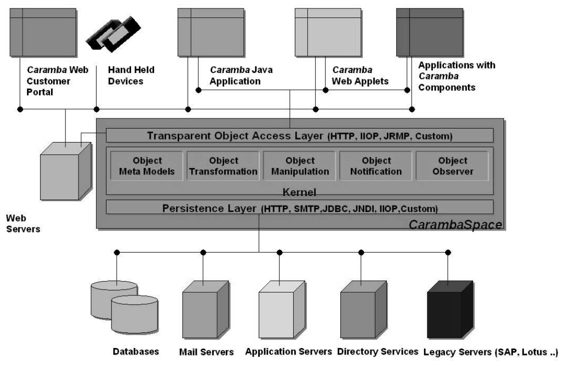


Figure 2. Conceptual Architecture

Figure 3 provides an overview of the client components, their connection to the CarambaSpace as well as the connection between a CarambaSpace to the persistent data store (e.g. embedded Database or relational DBMS).

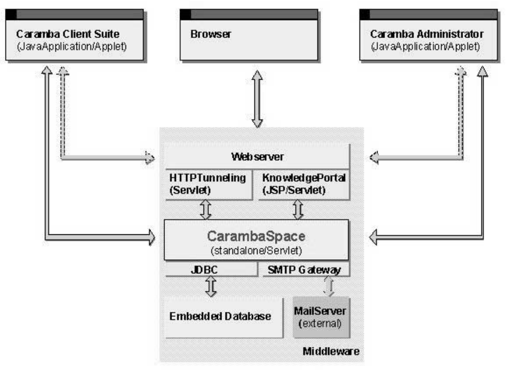


Figure 3. Caramba Overview

4. Components

Caramba's administrator components allow customization and extension of Caramba. The administrator toolset, as summarized in Table 1, comprises a set of components, which allow modification of the default meta-model (e.g. organizational model of project teams) and integration of company-wide information systems such as SMTP or Web server.

Table 1. Administration components

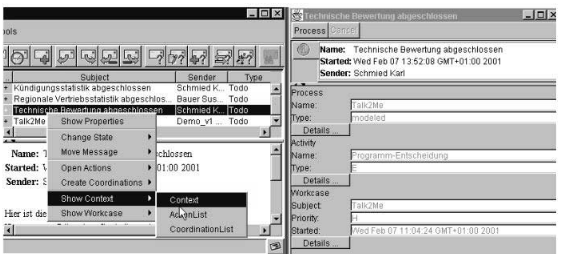
Meta-Model Administrator	manage meta-models (customization)
JDBC Wizard	integrate data from different datatabases
Relation Wizard	describe relations between objects
Service Administrator	administrate Caramba services
Mail Integration	specify email integration and forwarding
Web Server	Web Server (can be exchanged by custom web servers e.g. Apache, Netscape, IIS, etc.)

Caramba offers a suite of software components for end-users to support collaboration, coordination, and cooperation for distributed project teams. Table 2 provides an overview of the client components.

Table 2. Client components

ActivityCenter	manage work items and coordinate, cooperate, communicate with others
ObjectCenter	access, browse, and link objects
ProcessModeler	model and view business processes
ActivityAnalyzer	analyze and track work and project progress
Notification Center	register and manage object notifications
Knowledge Portal	access real time project data through the web

The *ActivityCenter* is the main "collaboration hub" for project team members. Here, project team members, work on their work items, route them to colleagues, and track the work item history if required. The collaboration model used is communications oriented. This means that Caramba users actively route work items to other team members, integrate artifacts into the system and link them with their activities. The ActivityCenter allows *continuous traceability* of business processes to team members, as depicted in Figure 4.



Subject	Sender	Type
Kündigungsstatistik abgeschlossen	Schmied K...	Todo
Regionale Vertriebsstatistik abgeschlos...	Bauer Sus...	Todo
Technische Bewertung abgeschlossen	Schmied K...	Todo
Talk2Me	Demo_v1 ...	Todo

Technische Bewertung abgeschlossen

Name: Technische Bewertung abgeschlossen
Started: Wed Feb 07 13:52:08 GMT+01:00 2001
Sender: Schmied Karl

Process
Name: Talk2Me
Type: modeled

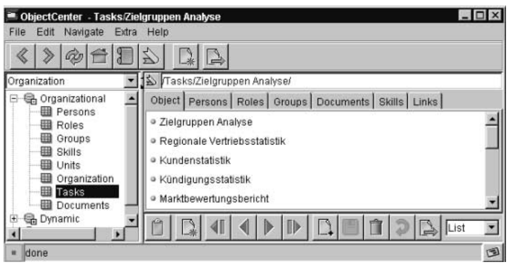
Activity
Name: Programm-Entscheidung
Type: E

Workcase
Subject: Talk2Me
Priority: H
Started: Wed Feb 07 11:04:24 GMT+01:00 2001

Figure 4. ActivityCenter – View Context

The second important end-user component, *Object-Center*, is depicted in Figure 5 and will be presented in this section. The goal of this component is to provide a mechanism to *link* (process) activities with artifacts, as discussed previously. Based on the meta-model discussed above, Caramba provides a set of *organizational objects*: Persons, Roles, Groups, Skills, Units, Organization, Tasks, and Documents (i.e. Templates). Utilizing these *organizational constructs* an administrator is able to model any organizational structure, such as hierarchical, flat, or matrix. Each object class consists of attributes describing the object. The *object class Persons* contains attributes about the Person such as name, address etc. The object class *Roles* allows definition of organizational roles such as "Head of IT". The object class *Group* defines project settings such as "Product Team IT-Solutions". *Skills* enable definition of required skill sets such as "Certified Java Developer". *Units* describe permanent departments such as "Marketing". The ObjectCenter provides means (by drag & drop) to link the rows of object classes with each other.

It also enables project team members to view relationships between *who* (organizational constructs) is performing *which* activities (Tasks) and using *what* (artifacts, documents). In order to fulfill the second design goal, namely to support the relationship between people, artifacts, and processes, Caramba supports modeling of business processes and their enactment by implementing a workflow engine (Process Modeler), utilizing the information presented in the section above (ObjectCenter), using tasks and their associated organizational constructs in directed graphs.



/Tasks/Zielgruppen Analyse/

- Zielgruppen Analyse
- Regionale Vertriebsstatistik
- Kundenstatistik
- Kündigungsstatistik
- Marktbewertungsbericht

Figure 5. ObjectCenter

5. Conclusions and Future Work

This paper outlined some design goals and an implementation of a system supporting mobility of context for project teams. The approach presented in this paper enables project teams to *link* knowledge *artifacts* such as documents (format independent) or database entries to process *activities* performed by human actors. These hyperlinks between artifacts and process activities enacted by people is currently not implemented by systems such as workflow management systems, project management systems, or groupware systems. We believe that supporting mobility of context is of paramount importance for distributed project teams. Our future work includes peer-to-peer modeling and execution of process models [e.g. 6], integration of distributed persistent data stores, and providing Caramba-functionalities as web services.

References

[1] W.M.P. van der Aalst, and A. Kumar, "A reference model for team-enabled workflow management systems", *Data & Knowledge Engineering*, Elsevier, 38 (2001), pp. 335-363.

[2] G.A. Bolcer, "Magi: An Architecture for mobile and disconnected Workflow", *IEEE Internet Computing*, May and June 2000, pp. 46 – 54.

[3] C. Bussler, "Enterprise-wide Workflow Management", *IEEE Concurrency*, 7(3), pp. 32-43.

[4] Caramba Labs Software AG (2002) http://www.CarambaLabs.com

[5] F. Casati et al., "Developing e-Services for composing e-services", *Proceedings CaiSE 2001*, Computer Science Lecture Notes, Springer Verlag, 2001, pp. 171-186.

[6] Q. Chen et al., "Peer-to-Peer Collaborative Internet Business Servers", HP-Labs Technical Working Paper HPL-2001-14.

[7] N. Craven, and D.E. Mahling, "Goals and Processes: A Task Basis for Projects and Workflows", *Proceedings COOCS International Conference*, Milpitas, CA, USA, 1995.

[8] U. Dayal et al., "Business Process Coordination: State of the Art, Trends, and Open Issues", *Proceedings of the 27th VLDB Confererence, Roma, Italy*, 2001.

[9] C.A. Ellis et al., "Dynamic Change within Workflow systems", *Proceedings COOCS International Conference*, Milpitas, CA, USA, 1995.

[10] A. Hausleitner, and S. Dustdar, "*Caramba - Ein Java basiertes Multimedia Koordinationssystem*", In Erfahrungen mit Java. Projekte aus Industrie und Hochschule. Silvano Maffeis, et al. (Eds.), dPunkt-Verlag, Heidelberg 1999.

[11] D.L. Parnas, "On the criteria to be used in decomposing systems into modules", *Communications of the ACM,* 15(12), pp. 1053-1058.

[12] D.E. Perry, and A.L. Wolf, "Foundations for the study of software architecture", *ACM SIGSOFT Software Engineering Notes, 17* (4), 1992, pp. 40-52.

[13] Shaw, M., and D. Garlan, *Software architectures, perspectives on an emerging discipline*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[14] J. Puustjärvi, and H. Laine, "Supporting cooperative inter-organizational business transactions", *Proceedings DEXA 2001*, Computer Science Lecture Notes, Springer Verlag, 2001, pp. 836-845.

[15] Schal, T., *Workflow Management Systems for Process Organizations*. New York: Springer 1996.

[16] Workflow Management Coalition (WfMC), *Workflow Management Specification Glossary*, http://www.wfmc.org