Distributed and Parallel Databases, 15, 45–66, 2004

Caramba—A Process-Aware Collaboration System Supporting Ad hoc and Collaborative Processes in Virtual Teams

SCHAHRAM DUSTDAR dustdar@infosys.tuwien.ac.at
Distributed Systems Group, Information Systems Institute, Vienna University of Technology

Recommended by: Dimitrios Georgakopoulos

Abstract. Organizations increasingly define many business processes as projects executed by "virtual (project) teams", where team members from within an organization cooperate with "outside" experts. Virtual teams require and enable people to collaborate across geographical distance and professional (organizational) boundaries and have a somewhat stable team configuration with roles and responsibilities assigned to team members. Different people, coming from different organizations will have their own preferences and experiences and cannot be expected to undergo a long learning cycle before participating in team activities. Thus, efficient communication, coordination, and process-aware collaboration remain a fundamental challenge. In this paper we discuss the current shortcomings of approaches in the light of virtual teamwork (mainly Workflow, Groupware, and Project Management) based on models and underlying metaphors. Furthermore, we present a novel approach for virtual teamwork by tightly integrating all associations between processes, artifacts, and resources. In this paper we analyze (a) the relevant criteria for process-aware collaboration system metaphors, (b) coordination models and constructs for organizational structures of virtual teams as well as for ad hoc and collaborative processes composed out of tasks, and (c) architectural considerations as well as design and implementation issues for an integrated process-aware collaboration system for virtual teams on the Internet.

Keywords: process-aware collaboration, Workflow, Groupware, virtual teams, Knowledge Logistics, interaction management

1. Introduction

Companies operating in highly competitive markets have an intensive need for continuous innovation and faster time-to-market for their products and services. Those products and services are the result of complex and expensive business processes, often executed by "virtual project teams". Such teams can be characterized by having (geographically dispersed) team members collaborating on shared projects. Virtual team members are frequently embedded in different organizations and collaborate across multiple business processes, time zones, and locations. Involved processes are ad hoc and highly dynamic and require the interaction of many domain experts. Virtual team managers on the other hand demand facilities for management and control. The outcomes are highly knowledge intensive and therefore require sophisticated tools for capturing and managing knowledge capital within the enterprise, with partnering organizations, and with customers. Prospective process-aware cooperative tools are required to record, map, and manage processes involved in knowledge work.

Software systems such as Workflow Management Systems (WfMS), Groupware, Knowledge Management (KM), Process Modeling, and Project Management (PM) have been used to automate or to augment business processes in organizations. In recent years there have been considerable attempts to merge or to integrate some of the categories mentioned above [e.g. 5–8, 18, 25–27] based on the understanding that for achieving effective and efficient virtual teamwork flexibility and control need to be integrated. Future systems aiming at supporting collaborative knowledge work for virtual teams need to be process-aware [e.g. 4], cover intra-organizational as well as inter-organizational processes (e.g. product value-chains) considering to enact all activities associated in those processes on the Internet regardless of location (mobility) and regardless of devices used [e.g. 15]. Every day the need for creating and replicating collaborative and innovative processes of the organization rises. Solutions required for highly efficient and effective "Knowledge Logistics" (i.e. who does what, when, how, why, using which resources) [7] require novel conceptual abstractions and revisited metaphors for collaboration and coordination, as well as novel technological solutions, which go well beyond current collaborative software systems [11] such as Groupware [e.g. 8], Workflow [1, 2, 30, 31], Project [8]—and Knowledge Management [25, 29], which constitute a highly fragmented collaborative systems market (e.g. figure 1). The reason is that processes in virtual teams are highly integrated and flexible in respect to the associations between artifacts, resources, and processes and therefore do not adhere to boundaries suggested by traditional software systems mentioned above. In this paper we discuss some current shortcomings of approaches (mainly Workflow, Groupware, and Project Management) based on models and underlying metaphors. Furthermore, we present a novel coordination model including new coordination primitives and abstractions for virtual teamwork. In particular we analyze (a) the relevant criteria for process-aware collaboration system metaphors (b) models and constructs for organizational structures of virtual teams as well as for ad hoc and collaborative processes composed out of tasks, and (c) design requirements and implementation issues for an integrated process-aware collaboration system for virtual teams on the Internet. Our approach will be reflected in the light of industrial requirements and case studies.

The remainder of this paper is structured as follows: The next section briefly outlines a motivating example for virtual teamwork and provides an overview of our proposed approach based on a categorization of collaborative systems and their underlying metaphors, coordination models and abstractions. Section 3 discusses related work in categorization of collaborative systems and distills our contribution in relation to related work. Section 4 discusses the underlying conceptual and pragmatic issues for the design and implementation of process-aware collaborative work management systems. Section 5 presents architecture, design, and implementation of such as a system—Caramba, which started as a research project in 1997 and evolved into a commercial product in late 2001. Finally, Section 6 summarizes and concludes the paper.

2. Motivation and context

Consider the following simplified virtual (project) team example scenario as a basis for this paper: An IT consulting team consists of 25 team members from five companies being

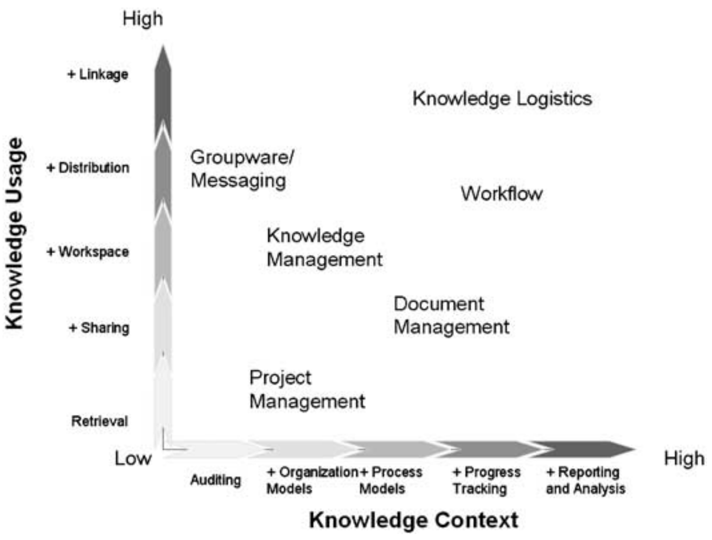


Figure 1. Conceptual technology framework.

responsible for IT systems consulting and implementation for many customers (projects). The virtual team consists of a project manager and team members with various responsibilities, tasks [e.g. 10], and skills. The program manager (managing multiple projects) and the project manager need to stay informed on all work activities and status information at all times. All team members have to work on customers' premises many hours a week and the project managers are travelling frequently as well. The presented scenario, although simplified, shares many characteristics with other virtual teams in many industries: Virtual team members (i) require *status information* on all work activities performed by other team members for a joint project (process-awareness); (ii) are increasingly *mobile*; (iii) *jointly work on artifacts* such as documents or databases (mostly asynchronously); (iv) require knowledge about the *multiple relationships* (associations or links) *between the artifacts* and the context in which they were created, shared, and distributed (i.e. who, what, when, in which context); and (v) team leaders require *on-demand access on project status and artifacts* as well as *critical communications* between team members and between team members and customers. Groupware systems, in most cases, follow a shared workspace metaphor (see next section) utilizing shared folders (e.g. web-based folders) and do not provide information on the *relationships between artifacts* and the *associated activities of business processes* (e.g. activity "presentation for customer Davis") or on the *status* of other team members' work activities. However, this relationship information is of paramount importance for knowledge-intense business processes of virtual 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 WfMS, due to the frequent ad hoc nature and multiple exceptions that would occur permanently.

To fully understand the context of collaborative technologies relevant for Internet-enabled process-aware collaboration systems, it is important to first analyze current systems. Our conceptual framework analyzes collaborative technologies along two orthogonal dimensions: Knowledge Usage and Knowledge Context as shown in figure 1. Each axis has a continuum of characteristic features.

Knowledge Usage describes the "paradigm" in which knowledge is used. In its simplest form knowledge is only retrieved. The next stage allows (in addition to retrieval) the sharing of knowledge, for example by using shared editor for synchronous joint editing. The following stage (includes the steps before), enables users to create workspaces by organizing knowledge artifacts using files in folder hierarchies. The distribution stage enables knowledge workers (in addition to the previous features) to distribute knowledge artifacts (objects) by using push/pull mechanisms. Finally, the link stage allows retrieval, sharing, workspaces, distribution, and in addition allows links between all knowledge artifacts. The second dimension *Knowledge Context* reveals contextual information on knowledge artifacts. Generally we can say that the higher contextual information is the more process awareness is stored together with artifacts. In its simplest stage it allows auditing of artifacts. For example users may view timestamp information on creation and routing of artifacts. The second stage enables organizational modeling, i.e. to define persons, roles, departments, and other organizational constructs required to design organizational structure. Organizational models allow organizations to define a set of access rights and rules for artifacts. The third stage additionally enables process modeling. Process tracking enables administrators to view the progress of business processes and the progress of activities as the building blocks of processes. Finally Reporting and Analysis supports analysis of all previously explained stages, and statistical comparisons between them. We find it useful to relate technologies on the market today to those two dimensions in order to elaborate our proposed approach. Groupware systems usually provide very low knowledge context information but provide relatively high knowledge usage capabilities, since they enable users to retrieve, share, organize their work in workspaces, and to distribute artifacts. Document Management systems are increasingly integrated with WfMS as recent mergers demonstrate (e.g. Lotus Notes/OneStone). Project management (PM) software is still mostly viewed as software for individuals (i.e. project managers) and rarely offers collaborative or business process aware solutions. Moreover, in most cases PM software is not integrated with corporate information systems and in fact is only utilized as a graphical modeling tool for outlining tasks. Most Knowledge Management (KM) systems on the market today are workspace-centered and provide only very simple forms to model organizational structures (e.g. using roles only, but not skills). To the best of our knowledge, few KM systems provide interfaces to business process modeling and enactment systems (the domain of WfMS) [19, 20, 30, 43]. Most KM systems enable users to retrieve artifacts from repositories (workspace metaphor), but rarely allow distribution and process awareness. Future work on the integration of Workflow and Groupware benefits from a process-oriented Knowledge Management approach, where Knowledge Management Processes (KMPs) define the interaction between knowledge works in a process-oriented manner and consist of activities that are supported by knowledge management key actions, such as searching, categorizing, and storing information [e.g. 29]. We suggest a new domain "Knowledge Logistics" [7], which—in addition to providing retrieval, sharing, workspaces, and distribution- allows for tight integration of artifacts and their relationships (e.g. data flow) (see Knowledge Usage axis). Furthermore, systems in this domain support reporting and analysis of communications and coordination patterns in a real time manner. In short, a process-aware collaboration system such as Caramba is one example of a Knowledge Logistics system and will be discussed in Sections 4 and 5.

3. Related work

Throughout the last 30 years, there has been a lot of work on classification models for collaborative systems, however, no "one and agreed upon" taxonomy of analyzing and understanding collaborative systems has been proposed so far. Academia and industry suggest various classification schemes. In industry for example, people frequently use the term e-mail and Groupware interchangeably. More generally, there is the tendency to classify categories of collaborative systems by naming a product (e.g. often many use the term Lotus Notes and Groupware interchangeably). Academic research has suggested many different classification models. For a recent extensive survey of collaborative application taxonomies see Bafoutsou and Mentzas [3]. DeSanctis and Gallupe [12], Ellis et al. [17] and Johansen [24] suggest a two dimensional matrix based on time and place, where they differentiate between systems' usage at same place/same time (e.g. electronic meeting rooms), same place/different time (e.g. newsgroups), different place/different time (e.g. Workflow, e-mail), different place/same time (audio/video conferencing, shared editors). This classification model helps to easily analyze many tools on the market today; however, it fails to provide detailed insights on collaborative work activities themselves as well as their relationships to business processes. Ellis [16] provides a functionally oriented taxonomy of collaborative systems, which assists in understanding the integration issues of Workflow and Groupware systems. This classification system provides a framework to understand the characteristics of collaborative systems and their technical implementations. The first category (Keepers) provides those functionalities related to storage and access to shared data (persistency). The metaphor used for systems based on this category is a "shared workspace". A shared workspace is basically a central repository where all team members put (upload) shared artifacts (in most cases documents) and share those among the team members. Technical characteristics of "Keepers" include database features, access control, versioning, and backup/recovery control. Popular systems examples include *BSCW* [e.g. 5], IBM/Lotus *TeamRoom* [23] and the Peer-to-Peer workspace system *Groove* [21]. The second category (Communicators) groups all functionality related to explicit communications among team members. Basically this boils down to messaging systems (e-mail). Its fundamental nature is a point-to-point interaction model, where team members are identified only by their name (e-mail address) and not by other means (e.g. by skills, roles or other constructs, as in some advanced Workflow systems). The third category (Coordinators) is related to ordering and synchronization of individual activities that make up a whole process. Examples of Coordinator systems include Workflow Management Systems. Finally, the fourth category (Team-Agents) refers to (semi-)intelligent software components that perform domain-specific functions and thereby help the group dynamics. An example for this category is a meeting scheduler agent. Most systems in this category are not off-the-shelf standard software. Both evaluation models presented above provide guidance to virtual teams on how to evaluate products based on the frameworks. Current systems for virtual teamwork have their strength in one or two categories of Ellis' framework. Most systems on the market today provide features for *Keepers* and *Communicators* support or are solely *Coordinator* systems (e.g. Workflow Management Systems) or are *Team-Agents*.

Combining workflow and groupware metaphors and primitives is not trivial. To our knowledge very few approaches exist today, which support highly dynamic processes such as in virtual teams or task forces [e.g. 8, 19, 24]. Well known, successful academic research has been undertaken in the area of adaptive workflow and research prototypes have been developed [e.g. 2, 9, 18, 27–30]. Adaptive workflow approaches allow for the dynamic modification of instantiated processes. Our approach is fundamentally different compared to approaches presented by many commercial WfMS and adaptive workflow systems, since our experience (grounded in many industrial case studies) is that most virtual teams begin to work on processes without modeling them in advance. The mechanisms adaptive workflow research prototypes such as Chautauqua [18], ADEPTflex [27], and WASA [30] build on is that single workflow instances can be adapted in *exceptional* cases. However, as far as virtual teamwork is concerned our industrial case studies (e.g. with consulting teams and new product development teams) show that "exceptions are the rule". Our goal is to provide a supporting environment (not automatisms) to solve "exceptions". Therefore, process remodeling or instance change propagation are not the preferred way of supporting virtual team members, who work in a loosely-coupled style and most of the time have no support from process modeling specialists. Hence Caramba currently does not provide automatisms if deviations from a modeled process occur and arrive at "inconsistencies" (compared to the modeled process) of the work case. The trail of all activities (control flow and data flow) is visible to team members and coordination primitives are provided to solve underlying problems in work activities. Similar requirements were presented in the area of crisis mitigation [19]. The CMI system [4, 19] is focused on task forces for crisis mitigation scenarios. Caramba and CMI share many requirements regarding flexibility for process templates. CMI provides similar concepts such as placeholder primitives for activities ("Tasks" in Caramba) and delegation principles (see Section 4.4). CMI supports the notion of process escalation, i.e. processes are dynamically extended and refined by the process participants. Caramba supports the notion of ad hoc and semi structured processes as well as combinations and does not require modeling of process templates before enactment. The notion of scoped roles provided by CMI has only limited support by Caramba and may serve as an area for future research. A detailed evaluation of current collaboration systems (academic and commercial) is out of the scope of this paper and the reader is referred to the literature [e.g. 3, 11]. From a functional perspective we suggest that a process-aware work collaboration system for virtual teams needs to (a) provide organizational constructs (e.g. persons, roles, skills, groups, tasks etc.) in order to flexibly model an organizational structure and responsibilities of virtual teams; (b) to provide constructs for modeling generic tasks and associated document-templates or applications in order to enact them for particular team members; (c) to provide the means to graphically model a control flow for business processes on a high granularity. From a technical perspective two requirements are fundamental for virtual teams: (i) cross-organizational process enactment for ad hoc and collaborative processes including analysis of interaction patterns between team members; and (ii) integration (and communications-references) of database repositories as an important resource for artifact management. We designed and implemented the above suggestions. Our goal for the next section is to discuss the conceptual foundations and required constructs, to outline architectural considerations, and to present some issues on design and implementation of *Caramba* [7].

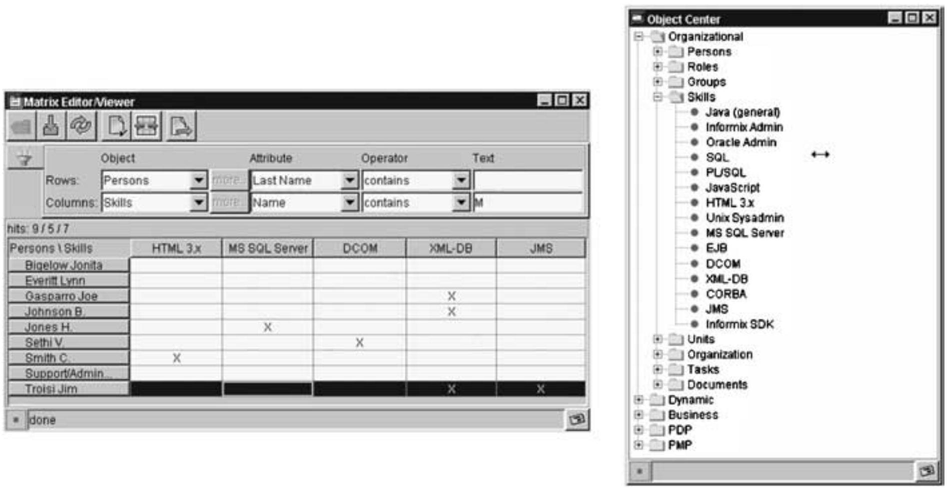
4. Process-aware collaborative work management

4.1. Team organization

Process-aware collaborative work management requires structural as well as process-oriented information. Based on the five aspects depicted above we have implemented Caramba, which allows virtual team members to fulfill the outlined activities. Three object categories build Caramba's core: Organizational-, Dynamic-, and Business Objects (summarized in Table 1). *Organizational Objects* enable the modeling of organizational structures and responsibilities for virtual teams and consist of the following objects categories: Persons, Roles, Groups, Skills, Units, Organization, Tasks, and Documents. Each object may be associated with all other objects. The reason to introduce multiple grouping constructs is that virtual teams based in various vertical industries have different interaction

Table 1. Object categories and their characteristics.

Objects	Characteristics
Organizational Objects	
Persons	Properties of persons involved in a virtual team.
Roles	Properties and Descriptions of Roles in the form of responsibility inside the organization (e.g. Manager of R&D, Marketing Director).
Groups	Properties of Groups (e.g. department names or informal working groups or task forces).
Skills	Properties and descriptions of Skills (a) found in the team or (b) required to fulfil tasks.
Units	Properties of organizational units (in some organizations equal to department names).
Organization	Properties of the organization.
Tasks	Properties of "reusable" tasks (e.g. write an offer, call customer, write specification).
Documents (templates)	Properties of document templates (e.g. Word, Excel, or any application or URI) associated to Tasks.
Dynamic Objects	
Processes	Properties of modeled business processes templates (design- time) including time and cost related information.
Workcases	Properties of instantiated (run time) business processes including knowledge trail of who, what, when, why, and how.
Business Objects	Database tables included by the organization to be utilized in coordination of work activities (e.g. product or customer database).



Object	Attribute	Operator	Text
Rows: Persons	Last Name	contains	
Columns: Skills	Name	contains	M

hits: 9 / 5 / 7

Persons \ Skills	HTML 3.x	MS SQL Server	DCOM	XML-DB	JMS
Bigelow Jonita					
Everitt Lynn					
Gasparro Joe				X	
Johnson B.				X	
Jones H.		X			
Sethi V.			X		
Smith C.	X				
Support/Admin...					
Troisi Jim				X	X

done

- Organizational
  - Persons
  - Roles
  - Groups
  - Skills
    - Java (general)
    - Informix Admin
    - Oracle Admin
    - SQL
    - PL/SQL
    - JavaScript
    - HTML 3.x
    - Unix Sysadmin
    - MS SQL Server
    - EJB
    - DCOM
    - XML-DB
    - CORBA
    - JMS
    - Informix SDK
  - Units
  - Organization
  - Tasks
  - Documents
- Dynamic
- Business
- PDP
- PMP

Figure 2. Organizational modeling of virtual teams—MatrixEditor and ObjectCenter.

patterns which are reflected in structural constructs. The end-user component *MatrixEditor* allows organizational modeling by choosing two object categories and establishing a relation as depicted in figure 2. The relationship can either be a predefined relationship symbolized with an “X” in the cell or modeled with a “named identifier” (e.g. “only basic knowledge of”).

The *ObjectCenter* component provides views on objects and their properties and enables to view associations between a selected object to other objects mentioned above. The possible relationships of selected objects are visualized by in the Object Center and according the organizational model defined in the meta model (see Section 5.3) they are visualized in the end user component. The model can be customized depending on the requirements and access rights of the virtual team member.

In some vertical industries team members only, for example, require *Groups* and *Roles*. In other industries, e.g. in virtual teams working in human resources organizations, the *Skills* construct is a benefit, since team members can coordinate their work by sending a work to be done to *Skills*. Based on many interviews, which are out of scope of this paper, we devised a model which provides constructs for most virtual teams: A *Person* may have multiple *Accounts* (to access the systems’ workspace), be member in many *Groups*, have many *Skills*, be responsible for many *Tasks*, have many *Roles*, and work in many *Units*. Each Task may be associated with (n)one or many *Templates* (e.g. Applications). Currently modeling of organizational hierarchies for virtual teams is limited. It can only be modeled using a “named identifiers” for the relation. As a minimum requirement, a project manager or responsible administrator will first define who (Persons) works with this project (process). Furthermore, other constructs (e.g. Roles, Skills, Groups, etc.) may be configured. By utilizing Organizational Objects, team members and managers will have more access to contextual information generated during enactment of processes. *Dynamic Objects* provide information on processes (process templates) and work cases (enacted process templates) including their properties, Organizational Objects (e.g. Persons) working on them, used artifacts, as well as time and cost information. *Business Objects* provide the option to include database tables of corporate information systems in order to use them in collaborative work activities (e.g. refer to a particular product in a database as an attachment and asking a consultant on more information on that product).

4.2. *Processes and activities*

Business Processes are a fundamental part of collaborative work and require thorough management. A business process can be defined (design-time) by a directed graph of connected activities using process modeling languages such as Petri-Nets or UML-Activity diagrams. Activities constitute pieces of work that form logical steps within processes [30]. They may be manual or automated in nature. Activities are performed (enacted) by one or many actors. Actors themselves may be persons or machines. In this paper our focus is on human actors. In order to reach business goals, activities need to be executed. Most WfMS impose ACID (Atomicity, Consistency, Isolation, Durability) properties on them, i.e. that activities are considered to be atomic and either carried out completely or not at all. Unfortunately, in most WfMS the user is forced to use either all or none of the ACID functionality. In this paper we propose a far less rigid approach. However, in most real-world collaborative work scenarios activities in virtual teams, which are routed between actors of one process, can be non-atomic. Georgakopoulos et al. [20] raise this issue stating that the ACID transaction model enforces task isolation, i.e. it does not permit task cooperation. This aspect is fundamental to understanding the way virtual teams coordinate and collaborate on joint activities. We refer to those pieces of activities being routed to other actors of the process as work items. Generally speaking, we can distinguish between modeled, semi-structured, and ad hoc processes. In modeled processes the information flow (control flow) between activities can be modeled *before* execution (instantiation or enactment). However, sometimes collaborative work processes cannot be modeled in advance simply because the flow cannot be defined in advance. In those cases where the control flow between activities cannot be modeled in advance but simply occurs *during enactment* time (run time), we speak of *ad hoc processes*. In most cases we have studied, virtual teams mainly require support for (semi-structured) ad hoc processes. In Virtual Teams (e.g. in consulting teams) a general process flow on a coarse-grained level of granularity can be defined, however, not specified to more detailed levels. Similar experiences were reported in [19]. Caramba aims at supporting the whole process continuum ranging from ad hoc processes with *no* underlying process model to *modeled* processes, including combinations thereof. The main focus, however, is on ad hoc processes. This represents one of the most difficult challenges, both, technically and conceptually for the end user. For example, it should be possible for a virtual team to initiate an ad hoc process and, from one particular activity, to provide a link to a defined process model. Additionally, the system should also allow starting from a process template and to deviate from this model, simply by deleting activities modeled in the process template or by adding new activities from a given task library (see Table 1). In most cases virtual teams operate with

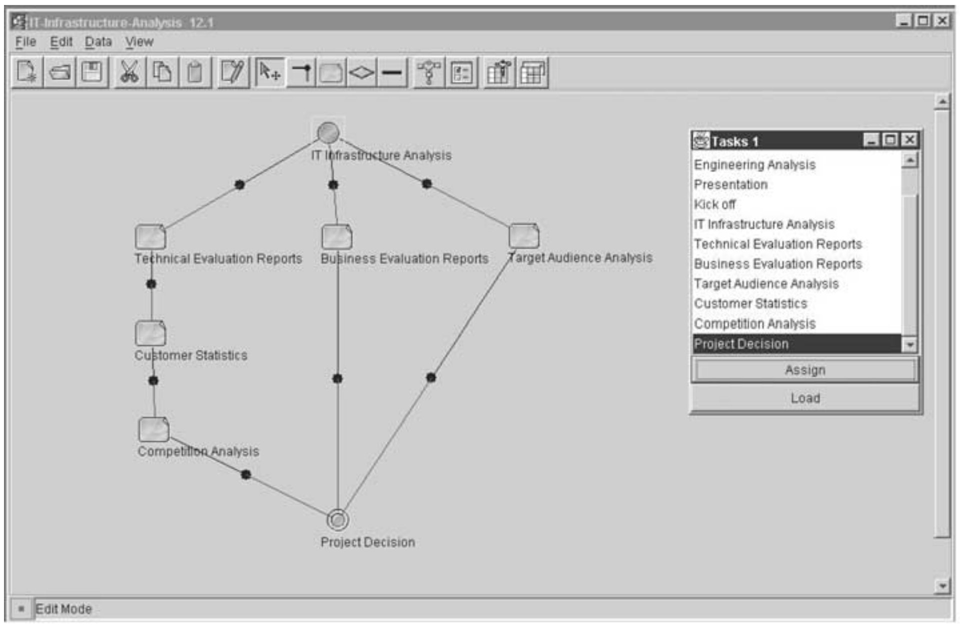
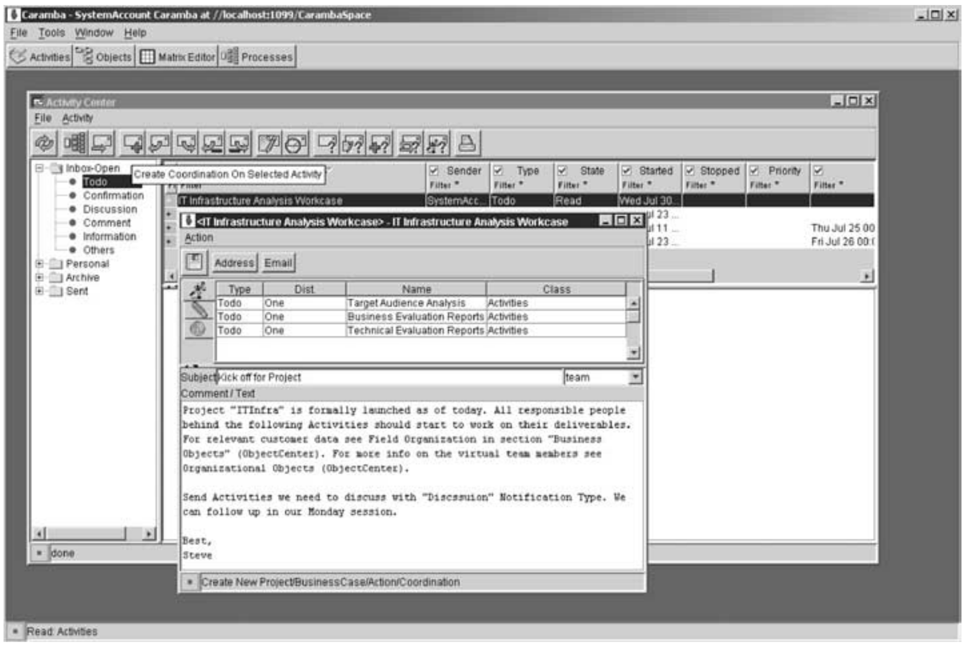


Figure 3. Process modeler.

ad hoc activities with some combination of modeled process templates. Those circumstances make it rather difficult, if not impossible, to apply traditional Workflow systems to the domain of virtual teamwork. In Caramba virtual team members do not need to model a process in advance in order to achieve process-awareness. The team members simply coordinate activities using the provided Organizational Objects constructs (for details see next two sections). For those cases, where a project manager is able to model a process template, a *Process Modeler* component is provided. It supports modeling of processes by building directed graphs consisting of "Tasks" (see Organizational Objects) and their relationships using the UML Activity diagram notation. When process templates are enacted (executed) we speak of *work cases*. Figure 3 depicts an example of a simple process template, which can be enacted in the course of the collaborative work efforts of the virtual team.

We distinguish between Tasks (see Organizational Objects) and activity instances. Tasks are entities on a generic (template) level, and when they are executed (instantiated) they become activity instances (referred to as activities, for short) and therefore are associated with Persons and other Organizational Objects. The intention is to model a business process template using generic task descriptions including their associated applications and/or document templates (i.e. invoked applications) and the associated Organizational Objects. Since modeled templates are generic they may be reused and instantiated by the owner of the "Begin-Activity" in the process template. In Caramba deviations from modeled processes (e.g. one Task can be skipped in a particular work case) do not require re-modeling and re-enactment of the process itself. The process participants may


Caramba - SystemAccount Caramba at //localhost:1099/CarambaSpace
File Tools Window Help
Activities Objects Matrix Editor Processes
Activity Center
File Activity
Inbox-Open
Todo
Confirmation
Discussion
Comment
Information
Others
Personal
Archive
Sent
Create Coordination On Selected Activity
Sender Type State Started Stopped Priority
Filter *
IT Infrastructure Analysis Workcase SystemAcc... Todo Read Wed Jul 30...
Thu Jul 25 00
Fri Jul 26 00:(
<IT Infrastructure Analysis Workcase> - IT Infrastructure Analysis Workcase
Action
Address Email
done
Create New Project/BusinessCase/Action/Coordination
Read: Activities

Type	Dist	Name	Class
Todo	One	Target Audience Analysis	Activities
Todo	One	Business Evaluation Reports	Activities
Todo	One	Technical Evaluation Reports	Activities

Subject: Kick off for Project — team

Comment / Text

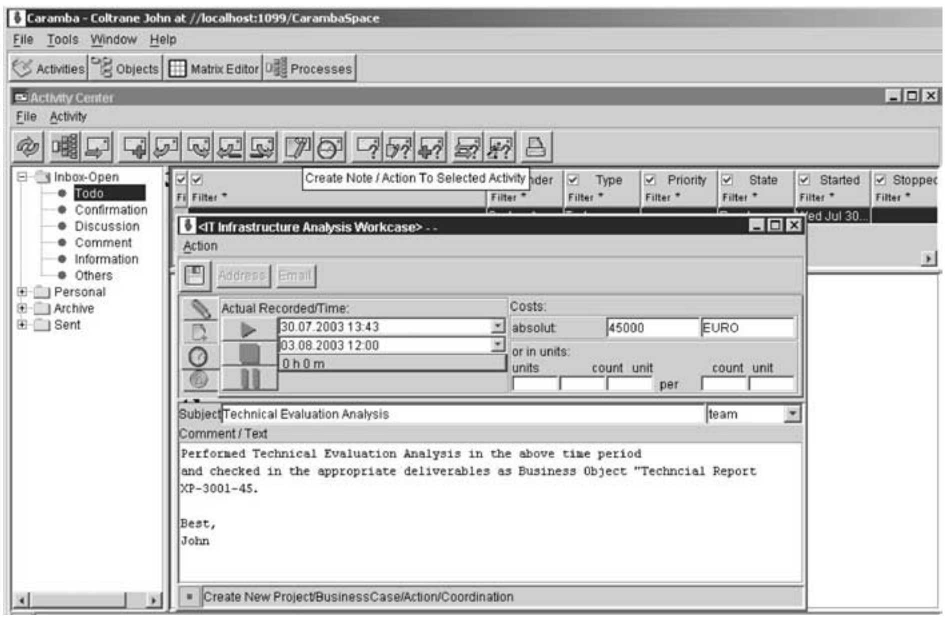
```
Project "ITInfra" is formally launched as of today. All responsible people
behind the following Activities should start to work on their deliverables.
For relevant customer data see Field Organization in section "Business
Objects" (ObjectCenter). For more info on the virtual team members see
Organizational Objects (ObjectCenter).

Send Activities we need to discuss with "Discssuion" Notification Type. We
can follow up in our Monday session.

Best,
Steve
```

Figure 4. Coordination of activity instances.

choose, depending on the context, to coordinate the activity instances to other Organizational Objects, without changing the underlying process template. Figure 4 depicts a scenario where the responsible person who initiated the process template sends the activity instance to the next steps. Caramba proposes the succeeding three activity instances to the user, who can agree with the proposed next steps or add/delete Organizational Objects as receivers. The overall idea is to provide process templates in the form of "best-practice process libraries" as part of the collaborative work management environment and to add flexibility to the team members in cases of exceptions to a process template. It is not required to model business process templates in advance in order to utilize process-aware information. When members of the system start (enact) a new process, which is not based on a modeled process template, an "ad hoc" process-template consisting of a "Begin-Activity" is automatically generated. The team members may use any Organizational Object to coordinate activities thereafter. This mechanism enables the system to keep track of all activities even without advanced modeling. To summarize the relationships between our proposed constructs: An activity may be non-atomic and may be composed of many work items (e.g. if the activity is split and routed to other team members). A work item may have many Actions. Actions are pieces of individually performed work on work items. In our above example two actions might be (a) invoking application XYZ and calculating the statistics, and (b) calling an expert and interpreting the results. In this case both actions can be stored and associated to the appropriate work item and activity respectively. We suggest that for a process-aware collaborative work management system


Caramba - Coltrane John at //localhost:1099/CarambaSpace
File Tools Window Help
Activities Objects Matrix Editor Processes
Activity Center
File Activity
Create Note / Action To Selected Activity
Inbox-Open
Todo
Confirmation
Discussion
Comment
Information
Others
Personal
Archive
Sent
Type Priority State Started Stopped
Filter
<IT Infrastructure Analysis Workcase> - -
Action
Address Email
Actual Recorded/Time:
Costs:
absolut
or in units:
units count unit per count unit
Create New Project/BusinessCase/Action/Coordination

Actual Recorded/Time: 30.07.2003 13:43 — 03.08.2003 12:00 — 0 h 0 m

Costs: absolut 45000 EURO

Subject: Technical Evaluation Analysis — team

Comment / Text:

```
Performed Technical Evaluation Analysis in the above time period
and checked in the appropriate deliverables as Business Object "Techncial Report
XP-3001-45.

Best,
John
```

Figure 5. Time and cost related information for actions.

it is relevant to capture the relationships between activities, work items and individual actions.

4.3. *Actions and scheduling*

Participants of processes receive activities (or work items) from other team members and are often required to work on them. For process-aware collaborative work management it is of paramount importance to retain all relationships between received activities and the personal work being performed based on the activities. We refer to this individual work activities performed based on work items as *actions*. One activity may have many actions associated with it. Actions themselves may be associated with document templates or applications which can be launched, when the user selects the appropriate activity, and selects the option to work on it (action). For each action time and cost related information may be entered, as depicted in figure 5. The reason for differentiating actions and work items is solely based on gained user experiences. It provides a “logical way” for end users to differentiate between a work item they receive and the (multiple) actions they perform based on or for that particular work item. Furthermore, it provides a “natural way” for users to select a particular work item and to view all actions they performed on that activity.

In some cases, team members also require scheduling their actions on selected activities. A process-aware collaborative work management system should enable the other team members to retrieve all relevant scheduling information of their colleagues in order to increase awareness of dependencies of work activities.

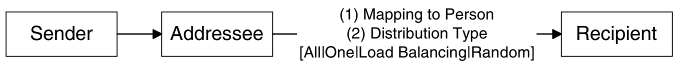


Figure 6. Coordination abstraction.

4.4. Coordination model and abstraction

Process-aware collaborative systems are based on a coordination model and a communication paradigm. The coordination model is the glue that binds separate activities into an ensemble [22]. Basically literature differentiates between three distinct paradigms: (i) Shared memory, (ii) Message passing, and (iii) Event-based. These approaches differ in the way they treat information exchange (communication). In shared memory approaches the communication takes place through read and write operations on a shared pool of data objects. Members of the shared memory space can access the edited data objects. This mechanism provides a comfortable abstraction for exchanging data, however, additional mechanisms need to be taken into account to provide notification to the receiver on information arrival or modification. In message passing systems processes or activities send and retrieve messages in order to communicate. A fundamental property of message-based systems is that the sender needs to know the receiver's address. Event-based systems follow a publish/subscribe pattern of interaction (metaphor). In this case processes and activities are required to subscribe (register) for particular objects and in case of changes on the subscribed objects, the event service sends a notification to the subscribed objects.

We follow a hybrid approach, which we will outline in the following paragraphs. As shown in Section 4.1, the underlying organizational model is known to all members of the system. This implies that the addresses of all objects (Organizational-, Dynamic-, and Business-Objects) are known to the system and have distinct addresses (shared memory). Our hybrid coordination model implements a messaging based system on top of the underlying shared memory. Figure 6 illustrates the coordination model abstraction.

The sender is required to identify an "addressee". The addressee may be any Caramba Object (Organizational-, Dynamic-, and Business-Object). For instance, the sender may send work to a *Role* or a *Skill* (or any Organizational Object) or to a process, i.e. its "Begin-Activity" (Dynamic Object). The sender may not be aware or in fact may even not care about who has the selected Role or Skill. Since the mapping between Addressees and Caramba Objects is $m : n$, the sender needs to select a *Distribution Type* (see Table 2) for the work being sent. Consider the following situation: Persons A, B, and C are members of Skill "Java

Table 2. Distribution types.

Distribution types	Characteristics
One	Activity instance is sent to one addressee, i.e. the first available who is online.
All	Activity instance is sent to all members of the class addressee.
Load balancing	Activity instance is sent only to certain members of the addressees, based on a defined load balancing algorithm.
Random	Activity instance is sent to a random member of the addressee list.

Table 3. Coordination primitives and semantics.

Types	Characteristics
Ad hoc coordination	Creates an ad hoc coordination based on selected activity instance.
Coordination	Sends (coordinate) activity instance to other member of the work team *after* completing working on the activity.
Forward	Sends activity instance to other members of the work team *before* working on the activity yourself.
Return	The activity instance can be returned to the sender. The receiver *does not agree* to work on the received activity instance.
Delegate	The activity instance can be delegated to other actors (e.g. Organizational Objects). The new receiver will be *responsible* for the activity instance.
Reply	Creates a reply for selected activity instance.
Action	Creates entries for (sub) activities instance based on selected activity. Actions may include information on created artifacts, time, date, and cost.

Programming" (they have the skill). The sender does not know who inside a project team has those Java programming skills (e.g. since this information is maintained by other people). However, since many persons may have this skill, a distribution mechanism needs to be specified by the sender. Choosing a Distribution Type [All | One | Load Balancing | Random] Caramba maps the Addressee to Recipient(s) (i.e. Person). This abstraction mechanism combines a shared memory approach with an explicit (user initiated) messaging approach.

Furthermore, in order to increase efficiency of collaborative work management we introduce *explicit* mechanisms describing means of coordination inside and between teams. Caramba provides a set of *coordination primitives* in addition to the well known such as *Reply* and *Forward.* For example, *Delegation* allows explicit delegation of work activities to other Organizational Objects (e.g. Persons, Roles, Skills, etc.). The coordination primitives are visible to all and the trail of activity coordination between team members is traceable for all team members involved in a project, depending, of course, on authorization settings. The coordination primitives and their characteristics are summarized in Table 3.

Traceability of associated activities, work items, performed actions including artifacts, and coordination primitives are of paramount importance. However, since many stakeholders work with this system (e.g. external partners, customers, freelance project partners, etc.) it is important to define coordination types for coordination primitives. Consider for example that the Sender (S) decides to use the coordination primitive *Delegation* for activity (A) which S sends to Recipient (R). The intention of S is to delegate a piece of work (an activity and its sub activities) to R. Since Caramba allows traceability of work activities team members are well aware of the fact that S delegated A to R. In this example the Coordination type *Team* is utilized. However, external partners should not see this delegation. If only S and R should know about this delegation, the coordination type *Private* would be used. If even outside Caramba users (in our example an external partner having access to the Caramba space via a portal) should be allowed to have access to the delegation information, the coordination type *Public* should be chosen. Table 4 summarizes the coordination types.

Table 4. Coordination types.

Coordination types	Characteristics
Public	Activity and related knowledge trail may be accessed from any Caramba user.
Team	Activity and related knowledge trail may be only accessed from users participating in the work case.
Private	Activities and related knowledge trail may only be addressed by sender and receiver.

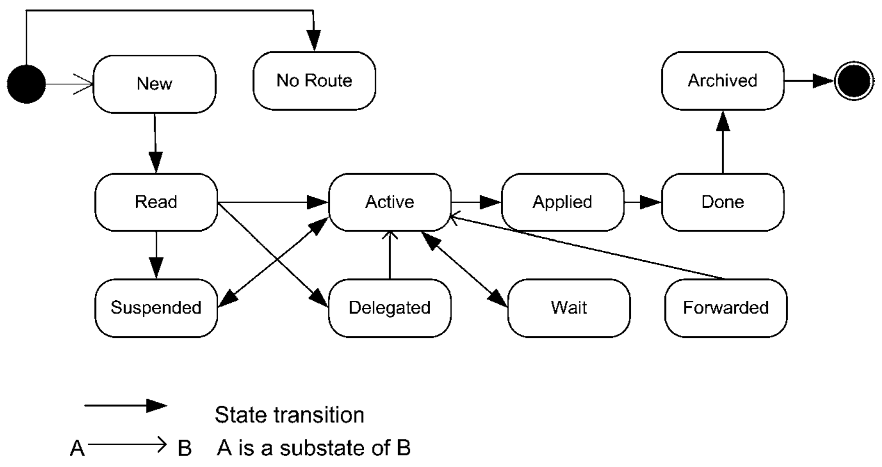


Figure 7. Activity state-transition diagram.

At any point in time activity instances have well defined states. Figure 7 presents a state-transition diagram for activities that is a graphical representation of activity states and their valid changes.

When a Task is enacted (instantiated) it becomes an activity (instance). Its first initial state is *new*. The *read* state shows that an activity has been "opened" in the user's work list and read. After having read an activity the user may change the state to either *suspended* (i.e. pausing work on the activity) or to continue working. When the user starts to work on an activity its state becomes *active* by explicitly changing the state so that team members are aware of the status. Activities which are delegated to other team members have the status *delegated*. The operational semantics of this state implies that the responsibility for the selected activity changed accordingly. Activities may be sent to other team members with the aim to keep them informed. The state for this activity is *forwarded*. The semantics of this state implies that the responsibility for the activity did not change and still remains with the sender. As the user starts to invoke applications associated with an activity its status changes to *applied*. In those cases where the activity cannot be routed to an addressee the *no route* state is triggered. One example this might occur is when activities are sent to non-Caramba users and the activity could not be delivered via the built in SMTP gateway. When work on an activity is finished the user has to set the state to *done*. This state will

Table 5. Notification types.

Notification types	Characteristics
To Do	Activity instance requires work from receiver.
Confirmation	Sender requires confirmation from receiver on the activity instance (e.g. signature for budget approval).
Discussion	Received activity instance requires further discussion (e.g. sender suggests that the activity needs to be discussed further (e.g. in a weekly team meeting).
Comment	Activity instance needs further comments from receiver.
Information	Activity instance is for information purposes only.
Other	Activity instance has none of the above notification types (e.g. activity is automatically converted from an e-mail and associated with the appropriate work case) into the knowledge trail.

trigger automatic coordination of the selected activity to the next process step (following a process template *if* it is a modeled process template). In this case the user may decide on the appropriate coordination primitive to be applied (see Table 3). When work on a selected activity is no longer required its state may be changed to *archived*. The activity will be automatically moved to a defined archive folder. All changes regarding states of activities are visible and traceable by the users of Caramba, if they are members of the appropriate project (process) and have the appropriate reading rights.

The coordination model we present in this paper is based on sending messages between Caramba Organizational Objects (e.g. Persons, Tasks, etc.). In order to increase efficiency of collaborative work, we introduce *notification types*, similar to those organizations use in office procedures. For instance, if sender S sends an activity to recipient R using the notification type *confirmation*, S expects a confirmation for that particular activity (e.g. signature required for purchase). Other notification types include *To Do*, *Discussion, Comment, Information,* and *Other*. The recipient receives the activities in the appropriate folder in his Caramba Inbox. In case S sends an activity with the notification type To Do, R receives it in his Inbox "To Do" folder. He then may decide to organize it according to different semantics (e.g. project name). By introducing notification types we can increase the semantics of activities being coordinated between team members. Notification types do not (directly) effect the states of activities (figure 9). State transitions are always triggered explicitly by users. Notification types help users to determine what is expected from them. The purpose of Notification Types is to increase efficiency in virtual teamwork by enhancing semantic information of work activities. Activity states enable team members to view on work items and their complete trail. Table 5 summarizes the supported notification types.

5. Design and implementation of Caramba

5.1. Software architecture

The goal of this section is to provide a brief overview of the design goals and to outline architectural considerations of *Caramba* [13–15]. 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. The main design goals encompass tight integration of many Workflow, Groupware, and Project Management features outlined in previous sections and to provide means for linking processes with artifacts and resources. Therefore support for meta modeling the team structure as well as the ability to integrate "Business Objects" (e.g. DBMS-tables such as product databases) 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. The software (middleware and client) is written in Java based on JDK 1.4 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*. For this it is important to decompose a system into a well-defined set of components that have clear responsibilities. Since architectures for virtual teams have to integrate with various corporate information systems installed in organizations, we decided to strive for a middleware style rather than a classical client-server style. Caramba services are hosted on a Caramba middleware (server). Clients may access the project's team-space (CarambaSpace) 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 (see Section 5.3) to integrate corporate DBMS and other resources such as SMTP-mail servers and company web servers. The following descriptions will point out the respective architectural style used in a particular layer or component. The Caramba software architecture (depicted in figure 8) is composed of multiple layers: middleware, client suite, and a persistence store.

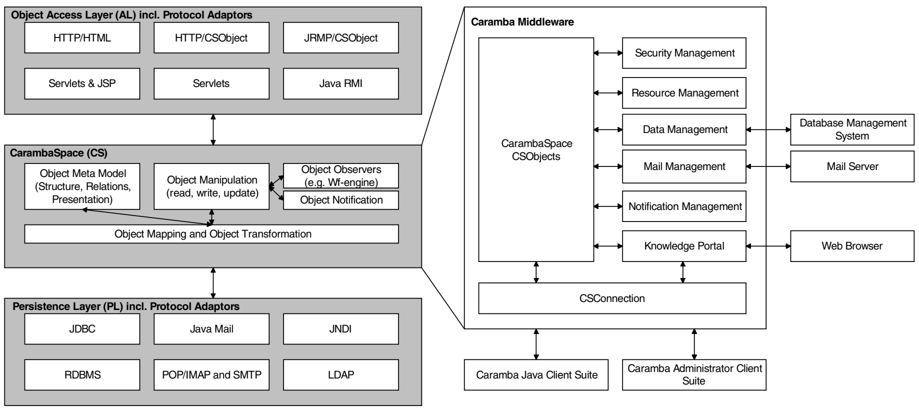

Object Access Layer (AL) incl. Protocol Adaptors
HTTP/HTML
HTTP/CSObject
JRMP/CSObject
Servlets & JSP
Servlets
Java RMI
CarambaSpace (CS)
Object Meta Model (Structure, Relations, Presentation)
Object Manipulation (read, write, update)
Object Observers (e.g. Wf-engine)
Object Notification
Object Mapping and Object Transformation
Persistence Layer (PL) incl. Protocol Adaptors
JDBC
Java Mail
JNDI
RDBMS
POP/IMAP and SMTP
LDAP
Caramba Middleware
CarambaSpace CSObjects
Security Management
Resource Management
Data Management
Mail Management
Notification Management
Knowledge Portal
Database Management System
Mail Server
Web Browser
CSConnection
Caramba Java Client Suite
Caramba Administrator Client Suite

Figure 8. Conceptual architecture.

Objects and services are accessed through the *Object Access Layer* (AL) from the CarambaSpace platform. 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 AL 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* (PL). Caramba leverages and adopts standard Java based technologies (e.g. JDBC, JNDI, HTTP, etc.) to access and integrate data. This architecture provides the required flexibility for flexible virtual teams where team members mostly connect using a web browser while still require access to all relevant context information provided by Caramba, such as who performed which activities (including sub-activities and time/cost information), which resources were utilized for this activities and how the activities are related to the overall business process. As stated earlier, Caramba provides a shared object space (CarambaSpace), which provides functionalities for fundamental operations and transformations required, as well as the communications needed with the other middleware components such as security management (authorization and access control), resource management, data management, mail management (i.e. integration and mapping of Caramba activities to e-mails), notification management, and a knowledge portal engine (providing servlets and Java Server Pages based access on a CarambaSpace). The detailed discussion of middleware components and their architecture would go beyond the scope of this paper.

5.2. Authorization and access control

A typical set up of a collaborative work management system for virtual teams has to cater for various account types to provide flexibility and security across organizational boundaries. Caramba provides an authorization and access control management component (Security Manager) for internal, administrator, partner, and external account types. The security management components allows (1) to manage and define access rights to all Caramba objects: Read (R), Insert (I), Update (U), and Delete (D); (2) definition of security roles (not to be confused with the Roles used in Organizational Objects) and the association between security roles and rights and (3) definition of the association between access rights and accounts. Access rights can be modified for each Caramba object by configuring the meta model. The database model allows not only human actors to have Caramba accounts but also software systems. The system provides four account types with a preconfigured set of access rights to Caramba objects: (a) Internal, (b) Administrator, (c) Partner (e.g. partnering organization such as supplier), and (d) External (e.g. freelancers, domain experts). Authorization is granted by a login which is encrypted, where the account login name and password have to be provided. Accounts may be activated with time stamps and their status can be retrieved and changed according to events.

5.3. Administration and user components

Caramba's administrator components allow customization and extension of Caramba. The administrator toolset comprises a set of components, which allow modification of the default meta model (e.g. organizational model of teams and their relationships) and integration of company-wide information systems such as databases, SMTP server, or web server. It consists of six components: (i) meta model administrator for managing and customizing the models (i.e. Caramba objects, their visibility, and the visibility of object attributes); (ii) JDBC wizard for database integration; (iii) relation wizard for modeling relations between objects; (iv) security manager for managing authorization and control; (v) mail integration wizard for specifying e-mail integration and forwarding; (vi) web server configuration wizard for specifying the web server to be used for web access to Caramba. A discussion of all administration components in more detail would go beyond the scope of this paper.

Caramba offers a suite of software components for end-users to support collaboration, coordination, and cooperation for virtual teams. The suite comprises Java applications and a web client. It consists of the following components: (i) ActivityCenter (see figure 4) for managing communications and coordination of work activities; (ii) ObjectCenter (see figure 2) for viewing and linking objects; (iii) MatrixEditor (see figure 2) for team configuration; (iv) ProcessModeler (see figure 3) for modeling processes; (v) ActivityAnalyzer for graphically analyzing interaction patterns in GANTT-like charts; (vi) NotificationCenter for registering and managing object notifications; (vii) Knowledge Portal for accessing the virtual team processes, artifacts, and resources using a web browser. The *ActivityCenter* is the main "collaboration hub" for project team members. Here, project team members, work on their work activities (including invoking applications, storing actions, scheduling their work, and providing time and cost information), route them to colleagues, and track the activity history if required (see Section 4.4 for underlying coordination model and abstractions). This means that Caramba users actively route activities 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. The *ObjectCenter* provides a mechanism to *view* Caramba objects and their relationships and to *link* activities with artifacts, as discussed previously. Utilizing the meta model tool, an administrator is able to model organizational structures for a virtual team and defining which Organizational Objects are required for his virtual team and which relationships are required. Each Organizational Object 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 to view all relationships of all Caramba Objects (i.e. Organizational-, Dynamic-, and Business Objects) and to link (by drag and drop) the rows of object classes with each other. It also enables project team members to view relationships between *who* (Organizational Object) is performing *which* activities and using *what* (artifacts, documents). In order to support the relationship between people, artifacts, and processes, Caramba supports modeling of simple process models and their enactment by implementing a workflow engine and modeling component (Process Modeler), utilizing the information presented in the section above (ObjectCenter), using Tasks and their associated Organizational Objects in directed graphs. For example the Caramba Knowledge Portal provides a detailed view on the "knowledge trail" on selected activities inside any instantiated process (i.e. work case) and the associated participants (virtual team members), their scheduling information and actions (if chosen to be visible by others) and artifacts (i.e. documents and business objects) used. This knowledge trail is dynamically built in real time and offers control flow and data flow information, including all work case artifacts and their relationships, providing added value to virtual team members. Due to space limitation not all components and their relationships can be discussed in more depth in this paper.

6. Conclusion

The contribution of this paper was to present conceptual foundations as well as design and implementation issues for process-aware collaborative work management systems. The system we presented—Caramba—supports virtual teams in their ad hoc and collaborative processes, by enabling *links* between artifacts (e.g. documents and database objects), business processes (activities), and resources (Organizational Objects such as Persons, Skills etc.). We discussed and applied a novel coordination model by introducing new coordination primitives and abstractions for virtual teams. We have presented the underlying motivation for process-aware collaborative work management systems for virtual teams (e.g. for a consulting team) and discussed the architecture and some implementation issues of Caramba aiming at improving process-awareness and traceability of collaborative work activities. Caramba was successfully used in many research groups as well as in commercial organizations throughout the last years. Our future work includes research on providing the presented functionalities using a service oriented architecture and support for loosely-coupled peer-to-peer scenarios for virtual teams.

Acknowledgments

All members of Caramba Labs are acknowledged for their role in the realization and the conceptual refinement of Caramba since 1997. I like to thank the anonymous reviewers for providing valuable comments and detailed suggestions helping to improve the paper.

References

1. W.M.P. van der Aalst, A.H.M. Hofstede, B. Kiepuszewski, and A.P. Barros, "Workflow patterns," Distributed and Parallel Databases, vol. 14, no. 1, pp. 5–51, 2003.

2. W.M.P. van der Aalst and A. Kumar, "A reference model for team-enabled workflow management systems," Data & Knowledge Engineering, vol. 38, pp. 335–363, 2001.
3. G. Bafoutsou and G. Mentzsa, "Review and functional classification of collaborative systems," International Journal of Information Management, vol. 22, pp. 281–305, 2002. Elsevier Science.
4. Baker et al., "Customized process and situation awareness," International Journal of Cooperative Information Systems, M. Papazoglou and G. Schlageter (Eds.), World Scientific, March 2002, vol. 11, nos. 3 and 4.
5. R. Bentley, W. Appelt, U. Busbach, E. Hinrichs, D. Kerr, K. Sikkel, J. Trevor, and G. Woetzel, "Basic support for cooperative work on the World Wide Web," International Journal of Human–Computer Studies, vol. 46, pp. 827–846, 1997.
6. G.A. Bolcer, "Magi: An architecture for mobile and disconnected Workflow," IEEE Internet Computing, pp. 46–54, May and June 2000.
7. Caramba Labs Software AG, 2002, http://www.CarambaLabs.com
8. K.C.C. Chan and L.M.L. Chung, "Integrating process and project management for multi-site software development," Annals of Software Engineering, vol. 14, pp. 115–142, 2002.
9. F. Casati, S. Ceri, B. Pernici, and G. Pozzi, "Workflow evolution," Data and Knowledge Engineering, vol. 24, no. 3, pp. 211–238, 1998.
10. N. Craven and D.E. Mahling,"Goals and processes: A task basis for projects and workflows," in Proceedings COOCS International Conference, Milpitas, CA, USA, 1995.
11. U. Dayal, "Business process coordination: State of the art, trends, and open issues," in Proceedings of the 27th VLDB Confererence, Roma, Italy, 2001.
12. G. DeSanctis and R.B. Gallupe, "A foundation study of group decision support systems," Management Science, vol. 23, no. 5, pp. 589–609, 1987.
13. S. Dustdar, "Towards integration of artifacts, resources, and processes for virtual teams," in Virtual Team: Projects, Protocols, and Process, David Pauleen (Ed.), Idea Group Publishing, 2003.
14. S. Dustdar, "Collaborative knowledge flow—Improving process-awareness and traceability of work activities," in 4th International Conference on Practical Aspects of Knowledge Management (PAKM 2002), December, Springer LNCS, 2002.
15. S. Dustdar and H. Gall, "Architectural concerns in distributed and mobile collaborative systems," Journal of Systems Architecture, Elsevier, vol. 49, pp. 457–473, 2003.
16. C. Ellis (Skip), "A framework and mathematical model for collaboration technology," in Coordination Technology for Collaborative Applications—Organizations, Processes, and Agents, Conen and Neumann (Eds.), Springer Verlag, 1998, pp. 121–144.
17. C.A. Ellis, S.J. Gibbs, and G.L. Rein, "Groupware: Some issues and experiences," Communications of the ACM, vol. 34, no. 1, 1991.
18. C.A. Ellis and C. Maltzahn, "The Chautauqua workflow system," in Proc. 30th Int'l Conf. on System Science, Maui, 1997.
19. D. Georgakopoulos et al., "Managing escalation of collaboration processes in crisis mitigation situations," in Proceedings of the 16th International Conference on Data Engineering, 2000.
20. D. Georgakopoulos, M. Hornick, and A. Sheth, "An overview of workflow management: From process modeling to workflow automation infrastructure," Distributed and Parallel Databases, vol. 3, pp. 119–153, 1995.
21. Groove, 2002, http://www.groove.net.
22. D. Gelernter and N. Carriero, "Coordination Languages and their significance," Communications of the ACM, vol. 35, no. 2, 1992, 97–107.
23. IBM Corporation, http://www.ibm.com.
24. R. Johansen, Groupware. Computer-Support for Business Teams, The Free Press: New York, 1988.
25. F. Maurer and H. Holz, "Integrating process support and knowledge management for virtual software development teams," Annals of Software Engineering, vol. 14, pp. 145–168, 2002.
26. G. Nagypal et al., "Integrating workflow and groupware functionalities for co-operating small and medium sized enterprises: A case study," in Proceeding of Seventh International Workshop on Groupware, IEEE Computer Society Press, Sept. 2001, pp. 38–43.
27. M. Reichert and P. Dadam, "Adeptflex—Supporting dynamic changes of workflows without losing control," Journal of Intelligent Information Systems, vol. 10, pp. 93–129, 1998.

28. G.D. Venolia, L. Dabbish, J.J. Cadiz, and A. Gupta, “Supporting Email workflow,” Microsoft Technical Report MSR-TR-2001-88.
29. R. Woitsch and D. Karagiannis, “Process-oriented knowledge management systems based on KM-services: The promote approach,” in Proceedings of the International Conference on Practical Aspects of Knowledge Management (PAKM), LNAI 2569, Springer-Verlag, pp. 398–412, 2002.
30. M. Weske, “Flexible modeling and execution of workflow activities,” in Proceedings of the 31st Hawaii International Conference on System Sciences, HICSS, 1998, vol. 7.
31. Workflow Management Coalition (WfMC), Workflow Management Specification Glossary, http://www.wfmc.org.