User-Defined Telecooperation Services

Volker Gruhn, Peter Herrmann, and Heiko Krumm
Universität Dortmund, Fachbereich Informatik, D-44221 Dortmund
{gruhn@ls10|herrmann@ls4|krumm@ls4}.cs.uni-dortmund.de

Abstract

A user-defined telecooperation service (UTS) provides service elements for application-specific communication and cooperation processes as well as integrated means for the service definition, adaption, and management. It supports user groups with particular communication, cooperation, and coordination needs which may change over time and which may be that special, that the service maintenance can be carried out only by the users themselves in an economic and satisfactory way. The users may be organized in various open and closed groups. They dispose of personal computing equipment connected via wide-area telecommunication networks. The users participate only from time to time. Therefore, there is a partial and varying accessibility of users and user sites. Interactions are mainly based on asynchronous communication operations. The cooperation and coordination functions have to consider unreachable users.

Thus, there is a looser coupling of users than in typical groupware systems which concentrate on the support of direct user interactions. Due to the loose coupling and the integration of mechanisms for the handling of unreachable addressees, however, UTS elements have a more detailed process structure than groupware interactions. Indeed, UTS elements may even perform cooperation processes with process and information schemes which can be compared with those supported by business process management systems. Nevertheless, in contrast to business processes, UTS elements are defined and managed by the users on their own, and the set of supported service elements will evolve during service operation time. For instance, UTSs may support special interest groups (e.g., software user groups or the members of an automobile club), teleworking, or interactive journals with direct reader-writer interactions. The set and lay-out of the service elements may follow the needs of a user community with growing experience.

A major problem of the service element definition by users is the mastering of concurrency. Due to their distributed execution, service elements will mostly define concurrent control flows for the sake of performance, while the design of concurrent behaviours may overtax non-experts. The problem is tackled by restrictions of control flow definitions accompanied by the provision of suitable basic coordination operations encapsulating necessary concurrent activities.

The paper introduces the notion of UTS. Fields of application are addressed. Moreover, we describe the principles of the service element definitions and outline the architecture of a supporting system.

Key Words: *Telecooperation service, value-added service, service elements, user interaction, user groups.*

1. Introduction

Value-added services are services which provide benefit to their users which go beyond the simple exchange of data. Typically, telecommunication providers try to set up value-added services in order to create traffic on their telecommunication infrastructure. The basic idea of a value-added service is to generate additional benefits by providing the right information to the right people at the right time. The basic assumption of our approach to user-defined, telecooperation services (UTS) as natural extension of value-added services is as follows:

- The broad availability of telecommunication services for a large set of users creates requests for many value-added services. These cannot be build in a central way by a large telecommunication company. This would neither be economical (because small user groups do not have the communication potential, the telecommunication providers are looking for), nor would it deliver solutions in due time. Thus, users should be able to define telecooperation services themselves. This means, predefined patterns of cooperation services must be defined, must be made acessible to end-users, and must be composable in an easy way. Only then, end users will be able to define their services on their own.

- User-defined telecooperation services change as frequently as communication and coordination needs of the participating people change. Thus, easy modification of these services is a key success factor for these services.

Starting from these assumptions we conclude that user-defined telecooperation services are value-added services which can be assembled easily (e.g. by and for a set of users, who have a particular communication interest in a given context) and which can be modified by their users without much administration effort. Examples of such services are:

- Teleworking and its administration: A set of people working from their home offices have certain synchronisation and communication needs.They have to exchange documents at certain points of time. They have to organize virtual (or personal) meetings, they have to agree on agendas for these meetings and they have to exchange minutes. All this can be supported by a UTS which supports the exchange of the information needed and which controls the flow of documents and information between the teleworkers involved. Other participants involved in this service (and taking benefit from it) are group leaders who can control progress by checking certain types of documents at certain points of time (to some extent even automatically), project managers who are supporting in the organization of meetings and budget controllers who can be supplied with budget relevant information as budget spent compared to work progress and similar information. All this is based on the UTS "teleworking and its administration" which sets up a context in which certain types of information are gathered, made accessible to certain users and actively forwarded to certain other users. This service does not only exemplify what a UTS looks like, it also illustrates that services are subject to change and have to be adapted to specific situations (after a while the budget controller may ask for additional information, work report formats have to be adapted to the standards of a particular company and so on).

- Automobile clubs: A completely different UTS is a UTS for the communication within an automobile club (as ADAC in Germany). The members of such a club usually do not communicate with each other directly. There is a periodic magazine which provides some information to the members, but there is not much further information exchange. A UTS which allows members of such a club to build special interest groups (for whatever subject they like) may be considered useful by the members (because they have the opportunity to get in touch with their co-members) and it may be useful for the club itself, because it may contribute to community building.

- Interactive journals: The UTS "interactive journals" provides all services needed for subscribing and accessing an electronic journal. In addition, this service provides support for submitting articles, letters to the editor, advertisement and call-for-papers to the journal. The service, moreover, asks the readers whether or not they agree to record which articles where accessed by them. If they agree they are allowed to access the list of all readers of these articles. Based on these lists discussion groups can be set up and an exchange of related information can be stimulated.

The notion of UTS as introduced by these examples has relationships to different areas. These related areas are:

- Groupware, business process management, telecooperation systems: The purpose of a UTS is to provide support for the communication and coordination of people in the context of a certain problem domains. This obviously relates to groupware systems which provide reactive support for common work [4], to business process modeling systems and workflow management systems which allow to describe business process and to use the process models to drive real processes forward [6]. Workflow management systems are particularly suited for highly-structured and routine-based business processes [21]. Telecooperation systems in general provide support for people located at various sites, but usually do not allow for the flexible definition of communication and coordination functionalities needed. Thus, UTS are related to groupware systems and workflow management systems by providing reactive and proactive communication and coordination support and by allowing user participation in the definition of this support. They are related to telecooperation systems by focusing on support for distributed workgroups. One of the most important and most recent requirement for workflow management systems is the support of distributed business processes [7]. The current approaches to handle distributed processes in the context of workflow management have to cope with typical distribution problems [14], which also have to be considered in the context of UTS. Some new approaches combine groupware functionality and workflow management functionality. In [20] it is described how multimedia-based communication is combined with business processes defined in terms of state transition diagrams. In this approach, strict process parts (for certain types of communciation) can be combined with flexible communication and coordination support for other process parts. The project *Item* which provides support for so-called complementary models focus also on this flexibility of communication and coordination [15].

- Modeling of distributed systems: UTS support distributed processes. UTS consist of service elements which are composed to build UTS. UTS are open systems (in the sense of ISO/OSI), whose implementation can be adapted dynamically to configurations of distributed networks. The flexible configuration and coordination of autonomous and interoperable software systems is subject to research in the recent years. An overview about languages for defining systems of this type and ways in which they may interoperate is given in [3, 22]. Moreover, the definition and the operation of UTS relates to distributed algorithms needed for communication and coordination control [16, 17].

In section 2 we discuss the example of building and supporting a software user group as an example of a UTS. We also discuss how such a UTS may be changed. Section 3 describes what a UTS definition looks like. In section 4 we explain the idea to compose complete UTS from service elements, thus allowing for an easy structuring of UTS. We discuss how new service elements can be defined. Section 5 gives a survey about UTS operation. It sketches the basic supportive service elements needed for UTS operation. Section 6 describes our experience in implementing a UTS platform. Finally, section 7 sums up our approach and concludes with an overview about the future implementation of UTS.

2. Software user group as an UTS example

In this section we discuss one UTS in more detail. This UTS is called "Software user group". It is meant to support the communication between different users of the same software and the communication between software users and the software provider. In our investigation of the basic functionality of such a service (compare to [8] for certain parts of it), we identified the following functional requirements:

- Release announcements: New software releases are announced. It is explained which new features are realized, how these can be put into operation and how they can be used.

- Customer care: From time to time the software users are interviewed in order to find out whether or not they are satisfied with the actual software releases. The results of these interviews (which can be carried out electronically by providing a form which is filled out by a user) are fed back into the product management.

- Organisation of user group meetings: User groups meet from time to time in order to exchange their experiences and in order to build alliances needed for putting some pressure onto the software supplier. These meetings help to improve the relationship between customer and the software used, it supports cross-selling (at least, if the software may be purchased as a set of components), and it helps to identify new development directions. The organization of these meetings depends on the subjects of interest, on the potential number of participants and on social events sponsored by the software supplier. The organization of such a meeting (inluding meetings of subgroups and setting up of subgroups) can be supported easily by asking the potential participants about their interests and requirements and by supporting their direct contact in advance.

- Problem tracking: Software bugs have to be reported to the software supplier. The software supplier has to check whether these bugs are accepted as bugs (of classified as new requirements), whether they have to be fixed immediately (in case of severe bugs) or whether they can be fixed in the context of a forthcoming patch. The result of this classification is forwarded to the customer who reported the problem. Within the UTS, customers can edit their problem reports by using a WWW interface. The incoming problem reports are fed into the appropriate processes at the supplier site.

- Workaround discussion: Users are provided with means to explain their way to use the software. This may concern the overall context as well as the use of certain workarounds needed for getting along with bugs or missing functionality.

- Customer subgroups: Customers are supported in building subgroups of the overall user group. A subgroup may, for example, be built by a set of customers who have certain requirements in common. Based on this commonality, they may decide to order a new component.

A UTS fulfilling these functional requirements is a reasonable starting point for the communication between a software supplier and his customers. In using this service certain requests for extending it were raised. One of these requests concerns the management of so-called red-flag bugs. A red-flag bug is a severe bug which demands that all users are informed about it. The idea is to avoid that this bug results in further problems at customer sites. By actively informing customers about it (instead of just letting them access the list if bugs identified), the UTS is modified. Other modifications of the UTS Software user group concern the way to communicate with experienced users who themselves reach an expert status. Once such customers are available and willing to function as an expert for certain parts of the software, parts of the communication may be redirected towards the experts. This modification is not due to an unexpected change of circumstances, but it is service immanent. Once service participants get more knowledgeable, their role in the service is updated and comunication links may be modified.

3. Service Definition

A user-defined telecooperation service UTS is provided to users cooperating and communicating in the broader context of a common area of interest. During operation, there may exist several concurrent cooperation processes which correspond to specific tasks. The processes are managed and controlled by the users themselves. The processes progress due to local actions of users and user-based application software operation. The telecooperation service contributes to the processes only indirectly. It provides for telecooperation service elements and for supportive services. The telecooperation service elements support the interaction of users and correspond to specific patterns of communication and coordination between users. The supportive services provide for global directory and storage capabilities. They are maintained by the provider in a reliable and permanent way. Moreover, there are several software components on the users' sites contributing to the operation of the telecooperation services. The service elements are executed locally by components providing for the man-machine-interface, for the local control, and for the communication with the service provider and other users. Additionally, there is a local object store acting as interface between telecooperation service elements and local applications. Finally, tools support the definition and modification of services.

Figure 1 depicts this basic structure of a UTS system. It emphasizes that the direct scope of a telecooperation service is restricted to the support of communication and coordination. All problem-oriented computations and decisions are under the control and responsibility of users and the effects of the telecooperation service elements are limited to the service interface and the local object store.

The set of users of a service can be structured into different groups and subgroups in accordance with special roles, capabilities, and interests. Users may enter and leave services dynamically. Also their group memberships may change in the course of time. Furthermore, there may be only temporary connections between users and service. Since users may be temporarily unreachable and may access the service via arbitrary data network access points, only users can initiate the establishment of connections to a service. Moreover, services have to concentrate on the support of asynchronous telecooperation interactions.

Besides of the user structure, the set of service elements contributes to the definition of a telecooperation service and the service elements may also be subjects of dynamic modifications. Service elements can be added and removed. Moreover, the scheme of a service element may be redefined changing the behaviour of the service element and the structure of its messages. Besides of these global modifications, there may be local service modifications on behalf of single users. A user can adapt the local presentation and handling of service interface events and their related messages to his special needs and preferences.

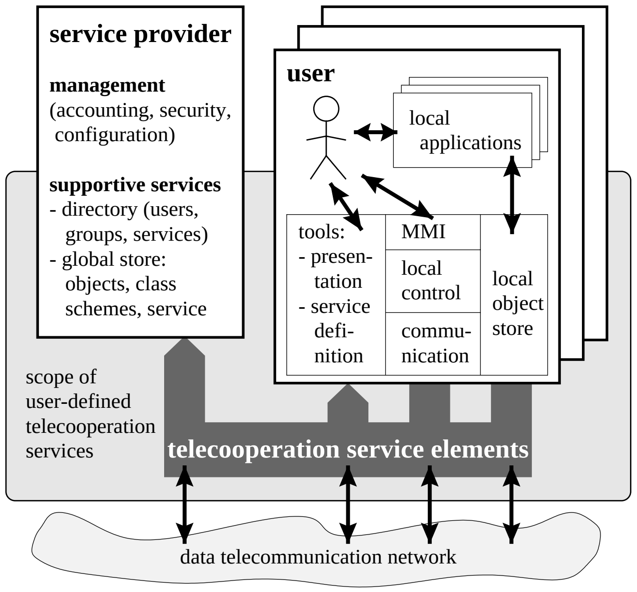
service provider
management
(accounting, security, configuration)
supportive services
- directory (users, groups, services)
- global store: objects, class schemes, service
user
local applications
tools:
- presentation
- service definition
MMI
local control
communication
local object store
scope of user-defined telecooperation services
telecooperation service elements
data telecommunication network

Figure 1. System structure

A service element is defined by its scheme which consists of a behaviour description and message schemes. The behaviour description defines the control and data flow of a service element execution in terms of those events which occur at the interface between service and user. As well-known from traditional telecommunication services there are user-initiated service interface events which stimulate the service (requests and responses) and service-initiated events (indications and confirmations). Each interface event transfers a message between service and user where the messages are object containers for two sorts of objects. User data objects support the communication between users only, i.e., users are the only sources, sinks, and interpreters of these objects and the values of the objects does neither influence the execution of service elements nor the behaviour of users. With respect to this, coordination parameter objects are used. The service implementation as well as users can create parameter objects which may influence the control flow of service element executions as well as they may represent user-relevant coordination information. The message schemes introduce the different message types and define their composition from user data and coordination parameter objects. Additionally message schemes contain presentation information. They describe the layout of messages and define static data (e.g., comments explaining the role of parameters).

4. Service Element Definition

Since most users are not computer professionals, the design of user-defined services may overtax the users. Referring to this we concentrate on two main problems. Firstly, users may overreach the service functionality. Secondly, the design of distributed service element behaviours may fail due to the particular complexity of concurrency. With respect to the first problem, we limit the scope of services to communication and coordination. Additionally, a service is composed from service elements and users can structure the service development accordingly. With respect to the second problem we restrict the concurrency in the control flow of service elements. The control flow of service elements, as it is has to be designed by users, is structured in accordance with few easy-to-understand principles. Since these restrictions can prevent efficient concurrent coordination solutions, we introduce particular building blocks for service elements which encapsulate predefined distributed functionality. Thus, the behaviour description of a service element scheme mainly defines a sequential control flow where the controlled operations correspond to basic communication and coordination activities, to modular building blocks encapsulating more complex activities, and to access operations to supportive services.

4.1. Control Flow

Basically the control flow of service elements is sequential or composed from sequential phases. Within a sequential phase three of the well-known control constructs of sequential programming can occur, the sequence, the alternation, and the case selection. There are no constructs supporting loops and jumps in order to prevent complex flow structures.

With respect to distribution, concurrency, and more complex control schemes following principles hold:

- Sequential control at initiator's site: Each service element execution is started by one request event. The request event starts a sequential flow of control which is managed locally at the site of the requesting user. Besides of the global supportive services the execution can access the object store which is local to this site.

- Sequential migration: The flow of control (and its management) can move to one other user's site in order to access the local object store there and to release the ressources from the former site's service element execution management.

- Subelements: Within a service element, the flow of control can request the execution of another service element. Recursive requests and migrating control flows are forbidden in subelements in order to prevent complex structures.

- Independent concurrent phases: The control flow may fork to concurrent phases which later on have to be joined again to one sequential flow at the same site under following restrictions: The phases themselves are sequential. Besides of one phase, which stays at the initiating site, all other phases migrate implicitly to other sites and stay there until termination.

The flow control constructs as well as the operations are represented by icons. On this basis, the behaviour of service elements is defined similar to program flow diagrams by means of a special graphical editor.

4.2. Interaction Operations

Besides of the flow control constructs, operations occur as nodes in a service element behaviour description diagram. There are access operations to supportive services and interaction operations. The interaction operations can be structured into three classes, communication, synchronisation, and coordination operations. While the communication and synchronisation operations basically have relatively simple execution schemes (e.g., one-way message transfer in the case of an unconfirmed communication), coordination operations mostly correspond to more complex distributed algorithms (e.g., distributed snapshot, cf. [17]). Common to all interaction operations are following aspects:

- Adressing and partner selection: There may be one or more peers of an interaction and we introduce a set of suitable addressing modes. Thus, e.g., a message may be sent to one specific receiver, to all members of a group of receivers, to exactly one member of a receiver group where this member has to be selected in accordance with specific criteria, to a subgroup of a receiver group where the subgroup has to have a minimal cardinality, or to the majority of a group. With respect to the selection mode we have to mention that it implicitly refers to the coordination operation of distributed selection.

- Temporary reachability of partners: Users may be temporarily not connected with the service and unreachable for an unknown period of time. Thus, interactions may be pending (e.g., messages wait for receipt, message senders wait for confirmations from unreachable receivers) and retard the progress of a service element. Furthermore, lots of pending but meanwhile irrelevant interactions may be queued for unreachable partners.

The definition of a service element attributes invocations of interaction operations by finite time-out values and default reactions in order to assure progress. The default reaction will also create a notification which is queued for the unreachable user and will inform him about the missed interaction. On the other hand, each user can define maximum life times for receive queue entries to prevent resource overflow.

Communication operations provide for the unconfirmed and the confirmed message transfer and are available in all combinations of the modes described above. For instance, if a user wants to transfer a binding order to exactly one member of a group, he may request a service element which mainly contains a confirmed message transfer under the selection addressing mode. Referring to our example (cf. section 2) a software user group may select and charge a representative (e.g., a person who will prepare the annual user group meeting) in this way.

Synchronisation operations coordinate the temporal progress of user activities. The realtime synchronisation operation supports timer based delay periods. In continuation of the last example (selection of a representative) a delayed inquiry can be established by realtime synchronisation and may ask for results of the representatives work.

The token operation forms a logical ring of addressees and transfers a token message in order to activate a user group and to serialize their actions. For instance, the representative might prepare a first proposal of the meeting schedule and send a circular letter with request for comments to the persons of the executive board of the software user group.

Finally, the lock operations support the fair allocation and deallocation of global lock objects. There are binary locks as well as counters which support the waiting for global conditions. Thus, our software user group may be in contract with the software supplier with respect to a fixed number of testing licenses, and the access to a license can be managed by means of a counter lock.

Coordination operations support the exchange of control information between users which enable them to coordinate their activities mutually in accordance with more complex patterns. With respect to following coordination operations we assume that the user activities manipulate the state of those objects which are local to the users:

- The snapshot operation sends requests for state copies and collects the replies. Moreover, it controls an implicit request propagation in order to ensure the consistency of snapshots. For instance, a snapshot which is established in the course of a free discussion phase will document a state of the opinion-forming process which is consistent in the sense that each opinion of the state is contained together with all of its influencing arguments.

- The synchronisation point operations support the arrangement of synchronisation points, the agreeing users are assumed to create local state copies. The reset operation supports the negotiaton of a binding reset point. As an example, by means of these operations the users of the software users group may realize a backtracking search of variants in order to improve the schedule of their annual meeting.

- The atomicity operation supports the user-based execution of a global two-phase commit process which ensures the atomicity of distributed activities. In our example scenario a subset of software users may conclude a quantity discount agreement with a software supplier.

Additionally there is the voting coordination operation which distributes questionaires, collects replies and evaluates their voting control parameters. In this way, our software user group may coordinate the conditions of their next annual meeting.

4.3. Supportive Services

Besides of the local object stores of the user sites there are following global supportive services which are maintained by the service provider:

- The user directory manages user groups, users, accounting, and security attributes of users.

- The service element directory manages the definitions of service elements including the behaviour descriptions and related message schemes.

- The service control store maintains state descriptors of service element executions and special control objects (i.e., locks, snapshot and synchronisation point descriptors).

- The security service provides for authentication, authorization, and encryption operations and performs the session key management.

- The global object store serves as centralized reliable storage for user data objects.

The operations of the first four suppportive services are accessed implicitly during the establishment of user connections and the execution of interaction operations, while write and retrieve operations of the global and local object store occur explicitly as nodes of the behaviour diagrams of service element definitions. Thus, one can specify the automated transfer of data between messages and object stores.

5. Service Operation

In the case of an open service we propose that a provider establishes an initial service and communicates its description and network addresses to potential users (e.g., via traditional communication media, via WWW, or via future service trader facilities). For closed services user organizations may start the service establishment and enter into an agreement with a provider. The initial service shall provide few and easy-to-understand basic service elements which correspond to the elementary communication and coordination needs of the potential users. Moreover, each initial service contains elements supporting the service operation and the user-based maintenance and modification of the service definition:

- The elements of the user connection management establish and release temporary connections between a user and the service. Also, an unknown user may establish a connection in order to run his enrollment.
- The user enrollment element creates a user account and results in initial group assignments of the new user. A further element supports the departure of users.
- The elements of the group management support the introduction, modification, and deletion of groups.
- The service element management elements support the insertion, retrieval, and modification of service element definitions. The definitions and the related behaviour and message schemes are communicated via messages and processed locally at user's site by the service definition editor tool.
- While the group and service element management elements provide for the basic access to directory entries, there is a needs for further support of service evolution. Therefore the evolution information service elements support the communication and agreement of service modifications. Users can post requests and proposals for group and service element changes. They can supply proposals under reference to requests and can initiate votings.

By means of these service elements users are assumed to form the service in accordance with their current needs resulting in a stepwise enhancement of the service.

In addition to the features mentioned, we already consider some extensions. One deals with the explicit support of object method definitions for objects contained in the messages and object stores of the service. Another interesting extension will be based on a notion of behaviour compatibillity and will support the subtyping of service elements in order to facilitate the service element modification.

6. Implementation

Presently there are two experimental implementations of UTS platforms. Both are based on Java [13]. The first platform was developed in 1997 [9]. It provides for user components and a server component which are Java-applications local to the user respectively server computers. The components interact via Java's remote operation call facility "Remote Method Invocation (RMI)" [18]. The argument and result objects of remote operation executions are exchanged in object containers which utilize Java's object serialization API. The platform provides for special user interfaces supporting the service management, service definition, and service operation. Their implementation is directly based on Java's window toolkit API "awt". The service elements are represented by sets of communicating finites state machines which can be adapted to user requirements by parameter settings. The free definition of new service elements is not supported. The project focussed on the feasibility of UTS generally and therefore besides of the platform two UTS applications were developed. They support distributed closed user groups. A teleteaching service supports the asynchronous internet-based communication of tutor and student groups. A group authoring application supports the edition of group reports. Both applications use the same set of basic service elements:

- Two-party communication elements support datagram-like notifications, confirmed messages, answer-and-question schemes, as well as the provision of proposal sets and the replies of the peers' choices.
- N-party-communication elements basically extend two-party communication schemes by means of responder group addresses and responder selection sets. Additionally there is the special service element "Discussion" where the initiating party manages the session, the membership in the group of participants, and the totally ordered forwarding of contributions.
- Cooperation elements support consensus and voting processes. There are simple votings, where participants can agree, disagree, or abstain. Other elements support the negotiation of appointments and the selection of subgroups, e.g., for the creation of car pools.
- Document transfer elements distribute documents, modification notifications, and access tokens.

The teleteaching application uses this basic element set to form specific service elements supporting the distribution of lecture notes and exercises, the coordination of training sessions, the free communication between participants, questions and replies between students and tutors, the management of examinations, and finally the creation of car pools for the visit of examinations. Additionally, the user management can be utilized to support the management of students, instructors, and courses in general, as well as it defines lecturers, tutors, and participants of courses. The group report application is oriented at a document preparation procedure which starts with a discussion resulting in an initial document and group structure. Several author and referee groups are instantiated. Subdocument responsibilities are defined. General guidelines, a timetable, and milestones are negotiated. Thereafter automated reminder messages support the in-time delivery of milestones. Subdocuments are processed locally. Subdocument export and import service elements facilitate the document management, which defines global document versions by means of distributed snapshots. Additional service elements concern intermediate discussions and negotiations as well as the preparation of physical meetings. Meanwhile, a medium-size computer network management company adopted the first platform and created an application which supports the communication and interaction of network management and maintenance personnel.

The current experimental UTS platform implementation extends the functionality of the first implementation with respect to two topics [19]. Firstly, it provides for a user interface which corresponds to that of a usual World Wide Web browser supporting the presentation and retrieval of hypertext documents. Secondly, it supports the relatively free definition and modification of service elements by users.

The implementation again is based on Java network programming technology. The platform consists also of user components and a server component which are Java applications local to the user and the server computers. Ideally, we would like to replace the user components by a usual Java enabled Web-browser which accesses the Web-pages of UTS applications. Applets contained in the pages process user interactions and cooperate with servlets of the corresponding UTS server component. The security restrictions of applets, however, disable the access to objects which are local to the user outside the browser environment. Therefore, we presently use the opposite approach. The user component is a regular Java application which integrates the HotJava HTML Component [10] supplying browser functionality. Thus, HTML-based Web-pages of UTS applications can be used also, and users can interact with a UTS via the well-known Web-interface.

A UTS application appears to a user by a set of Web-pages:

- Introduction and description pages outline the UTS application and provide for roadmaps and links.
- Whiteboard pages can disseminate short general dynamic informations.
- Service element selection pages offer the service elements available. The user can activate elements. The implementing JavaBeans will migrate to the user's site and process the execution of the invoked service elements.
- Service element execution management pages show the current status of service element executions relevant to the user. User's can transfer the local control to service element executions. Moreover, they can stop, continue, and abort executions.
- Service interaction pages are presented on behalf of the execution of service elements. They support the exchange of information between the service element execution and the user. (E.g., if the execution asks for a user message to be transfered to another user, there will be two interaction pages, one to input the message at the initiator's site, another later on, when the responding user receives the message.)

The implementation of service elements utilizes the software component approach JavaBeans [2], in particular Enterprise JavaBeans [5]. As new service elements can be constructed from an arrangement of already defined service elements, configurations of JavaBeans can form JavaBeans again. Besides of the runtime interface, JavaBeans support a design time interface. Application builder tools utilize the design time interface in order to explore properties of available beans and in order to customize the beans used. Beans can have a graphical interface which may be customized by the application builder tool, too. In particular, builder tools can have the character of graphical editors and we expect that future tools will provide for very comfortable and user-friendly bean definition functions.

Therefore, service elements are implemented by beans and we resort on builder tools for the purpose of service element definition and modification. Presently, we use the BeanBox tool which is contained in the current JavaBeans Development Kit [1]. Each of the basic elements introduced in section 4 is implemented by a corresponding bean. The beans are supplied with suitable graphical symbols supporting the editing of control and data flow graphs. In order to define a new service element, a user activates the BeanBox, instantiates basic service elements, and arranges their icons in accordance with the control and data flow of the new service element. Thus, basic service elements and control operations (which are all implemented by beans) form the nodes of a graph defining the new service element. Moreover, each node of the graph can be customized in order to adapt its properties. The properties mainly concern the user interface representations of the corresponding actions.

In consequence, two major parts of the user component of the UTS platform are implemented by re-use of general Java components: The BeanBox supports the definition and modification of service elements and the HotJava browser performs the over-all interaction between user and service. Additionally, there is a local object store component which manages local objects and their directories in order to support a gateway between local applications and UTSs.

7. Concluding Remarks

We proposed a new type of network application which supplies a flexible and comfortable support for user communities with more occasional and asynchronous cooperation needs. While there already exists a stable definition of the conception and also there are two successful implementation projects, we are planning to perform several more extensive usage experiments in order to gain practical experiences. Additionally, we will enhance the tool support and the implementation of supportive services to facilitate experiments with open and wide-spread user groups. In particular, a more user-friendly tool for the graphical service element definition than the Java BeanBox is necessary in order to support the service definition by non-expert users. This tool shall provide for a very comfortable user-interface. Furthermore, design errors should be detected by automated analysis and property checking functions. With respect to the supportive services, high-performance implementation architectures focussing on distributed server systems are of interest. Moreover, we plan the integration of security services.

References

[1] JavaBeans Development Kit. See reference in [12].

[2] JavaBeans — Component APIs for Java. Reference in [12].

[3] P. Ciancarini and C. Hankin (eds.). Coordination Languages and Models. In *Proceedings of the 1st International Conference on Coordination*, Springer LNCS 1061, 1996.

[4] C.A. Ellis, S.J. Gibbs, and G.L. Rein. Groupware — Some Issues and Experiences. *Communications of the ACM*, 1(34):38–58, 1991.

[5] Enterprise JavaBeans. See reference in [12].

[6] D. Georgakopoulos, M.F. Hornick, F. Manola, M.L. Brodie, S. Heiler, F. Nayeri, and B. Hurwitz. An Extended Transaction Environment for Workflows. *Distributed Object Computing, IEEE Data Engineering*, 16(2), September 1993.

[7] G. Graw, V. Gruhn, and H. Krumm. Support of Cooperating and Distributed Business Processes. In *Proceedings of the 1996 International Conference on Parallel and Distributed Systems, ICPADS '96*, pages 22–31, Los Alamitos, California, June 1996. IEEE Computer Society Press.

[8] V. Gruhn and J. Urbainczyk. Software process modeling and enactment: an experience report. To appear in *Proceedings of the International Conference on Software Engineering*, Kyoto, April 1998.

[9] M. Hehn and M. Rieks. Netzbasierte Kooperationsdienste und ihre Realisierung aus Java-Bausteinen am Beispiel eines verteilten Autorensystems und eines verteilten Fernstudiensystems. Diploma Thesis, University of Dortmund, FB Informatik, LS 4, 1997 (in German).

[10] HotJava HTML Component. See reference in [12].

[11] Java Remote Method Invocation Specification. Sun Microsystems, Inc., 1996.

[12] Java: Products and APIs. Available in WWW: http://java.sun.com/products/.

[13] Java Development Kit 1.1 Platform (JDK). See reference in [12].

[14] C.W. Loftus, E.M. Sherrat, R.J. Gautier, P.A.M. Grandi, D.E. Price, and M.D. Tedd. *Distributed Software Engineering*. BCS Practitioner Series, Prentice-Hall, 1995.

[15] Max Mühlhäuser. Modeling and Design of Complex Cooperative Software. In *Proceedings of the 1st IEEE Conference on the Engineering of Complex Systems*, Ft. Lauderdale, November 1995.

[16] Michel Raynal. *Networks and distributed computation — concepts, tools and algorithms*. North Oxford Academic, London, 1987.

[17] Michel Raynal. *Distributed algorithms and protocols*. Wiley & Sons, Chichester, 1988.

[18] Java Remote Method Invocation and Object Serialization (RMI). See reference in [12].

[19] Arkadiusz Speemann. Nutzerdefinierte Telekooperationsdienste implementiert auf Basis von Java-Beans. Diploma Thesis, University of Dortmund, FB Informatik, LS 4, 1998 (in German).

[20] Koji Takeda, Kazuo Sugihara, Mitsuyuki Inaba, and Isao Miyamoto. Modeling and simulation of Multimedia Interfaces and Agents for Flexible Working. *IEEE MultiMedia*, 3(2):40–50, 1996.

[21] B. Warboys. Reflections on the Relationship Between BPR and Software Process Modelling. In P. Loucopoulos (ed.), *Proceedings of the 13th International Conference on the Entity-Relationship Approach*, in LNCS 881, pages 1–9, December 1994. Springer-Verlag.

[22] P. Wegner. Coordination as Constrained Interaction. In [3], pages 28–33.