Enhanced User Support for Mobile Ad-hoc-Processes*

Klaus Haller, Michelle Ackermann, Claudio Munari, Can Türker
Swiss Federal Institute of Technology Zurich
Institute of Information Systems
CH-8092 Zurich, Switzerland
{haller,tuerker}@inf.ethz.ch
{mackerma, munaricl}@student.ethz.ch

Abstract: In ubiquitous application scenarios, the information (processing) needs of nomadic users often cannot be satisfied by a set of pre-installed processes like in common workflow systems. Instead, nomadic users require ad-hoc processes that are unique (personalized) and usually do not appear in the same way more than once. As a consequence, ubiquitous applications also require a sophisticated user management such that ad-hoc processes and users may find each other and interact dynamically, for instance, to implement/use location-aware services. In addition, ad-hoc processes must be able to dynamically adapt to changing conditions which often occur in ubiquitous application scenarios. We recently implemented user management, location-based services, and dynamic process changes as an integral part of the AMOR (Agents, MObility, tRansactions) system. This paper sketches these extensions of AMOR towards a prototype system for a peer-to-peer-based ubiquitous process execution environment with enhanced user support.

1 Motivation

Users benefit from ad-hoc processes and distributed implementations of their execution environment since for the first time information systems support users not only in standardized situations like workflow systems do. Instead, ad-hoc processes can provide support ubiquitously and in a 24/365 way for everybody. One possible example for such an ad-hoc process is planning an evening. Here, the user states its preferences (e.g. science-fiction movie, restaurant with french cuisine) and then the ad-hoc process reserves a table in the restaurant and a ticket for a movie. Also, it plans how the user could get to each place. Thus, this new generation of ad-hoc processes can support users in their every-day life where situations are unique and usually not appear in the same way more than once.

Executing ad-hoc process in a distributed way requires a sophisticated user management such that users and ad-hoc processes find each other and can interact. For example, if the user announces his preferences to the ad-hoc process in the morning, but not all reserva-

*This work is funded in part by the Swiss National Science Foundation within the project MAGIC.

tions can be made directly, the user might work on a different PC in his office when the ad-hoc process succeeds with the task during the day. Then, the ad-hoc process must notice this and contact the user in his office. In this way, ad-hoc processes imply shift in the usage of process technology: Whereas employees in their offices can be assumed to be static, this does not hold mobile users that cannot be assumed to be located at only one place. Consequently, ad-hoc processes respect the nomadicity of users by implementing location-aware services. For example, the current location of the user is then an input value for evaluating a time table information.

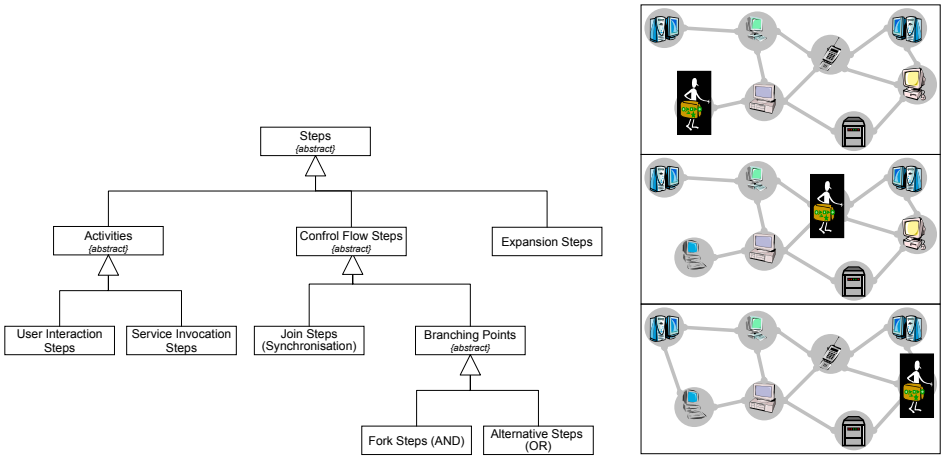
Besides, dynamic process changes are required for ubiquitous applications. In case of the evening planner, for example, the user might miss the metro and therefore not reach the cinema in time. Also, the ad-hoc process might figure out that there is no cinema in town. Both requires that the ad-hoc process is able to dynamically adapt, for instance with the help of an decision support system, such that nevertheless there is a valid plan constructed for the evening.

All these concepts have been incorporated into the AMOR system within two semester works [Mun03, Ack03]. The AMOR framework [HST03, HS03a, HS01] executes ad-hoc processes in a peer-to-peer style way without any centralized component. Its speciality is the reliable execution based on the transaction properties, which are implemented in a truly distributed way.

In this paper, we sketch the extensions of AMOR. Section 2 briefly recapitulates the basics of the system architecture and the ad-hoc process execution. Section 3 presents the user management and how ad-hoc processes can interact with the user. This user management is additionally the base for location-aware services in Section 4. Closely related to nomadic users and to dynamic environments is the support for modifying ad-hoc processes, which we discuss in Section 5. Section 6 concludes this paper with a short summary.

2 The AMOR-System

AMOR is a prototype system for the execution of ad-hoc processes. The difference between ad-hoc processes and workflows is mostly on a conceptual level. Workflows are well-defined, executed in 'controlled' environments like companies, and are executed many times. Ad-hoc-processes are put together 'ad-hoc' to satisfy a user need which might not emerge again. This requires a different system architecture, but the **process definition** is quite similar to the one of workflows. Ad-hoc-Processes are composed of a set of steps. These steps can be of different types: *activities*, *control flow steps*, and *expansion steps* (see Figure 1(a)). Activities are steps which contact and/or influence the outside world, e.g. they present data to the users or allow them to give information to the process (*user interaction steps*). On the other side, a *service invocation step* invokes services on one or more peers, which change or retrieve information, e.g. processed by databases, sensors etc. An important difference to e.g. RPCs [GR93] is that the service invocations are not hard-coded but specified in a predicative way [HS01]. Control flow steps define alternative execution paths in the process (*alternative steps*) or parts of the workflow which can be



(a) Process Steps (b) P2P Process Execution

Abbildung 1: Ad-hoc-processes

executed in parallel (*fork steps*). Such parallel paths come together in *join steps*. Finally, there are expansion steps. They define situations in the process, where expansion by additional steps is required.

AMOR implements a *peer-to-peer process execution approach* for ad-hoc processes. As Figure 1(b) illustrates, an ad-hoc process instance moves from peer to peer in order to invoke services on the corresponding peers. Thus, ad-hoc processes are implemented as mobile agents and do not rely on a centralized engine. Sample services in our prototype are for instance a time table information system, booking cinema tickets, or reserving tables in a restaurant.

To realize such an execution model, each peer has a local coordination layer (Figure 2). This layer maintains meta-data required by ad-hoc process to invoke local services. This meta-data management is implemented by different repositories. Firstly, a service repository governs information about the locally available services. Secondly, the network repository is responsible for the connections to other peers, by which the different peers establish a peer-to-peer network between them. This combination of a (local) service repository and the network repository allows the processes to discover services locally as well as remotely (see [HS01]) for technical details). To speed-up the service discovery in case of similar service queries, information regarding services on other peers could also be cached locally.

The service query functionality is required by the service querying components of the coordination layer of the ad-hoc processes. The latter contact the local coordination layer when they invoke steps and therefore require service providers. So each ad-hoc process (certainly) must have a process description to know its application semantics, e.g. which

service provider it needs. Also, each process manages its execution state such that the coordination layer of the process to controls the actual process execution on its own without requiring a centralized engine.

For exchanging information within the process, the process coordination layer provides additional components, e.g., a *process board*. It manages variables – objects addressed by their names – of the process. Thereby, it realizes information flow between different – sequential or parallel – steps of a certain process, because each step has read and write access to the process board.

In case of service parallelism, clones of the process independently invoke semantically equivalent services of different peers, before they later synchronize the clones by using an activity board. This *activity board* is again a set of objects, but it can only be accessed by clones of the same activity.

Additionally, the processes and the peers have *process synchronization components*, which enforce the isolation, one of the ACID properties. This is an important outcome of the AMOR project and was published in [HS03a, HS03b].

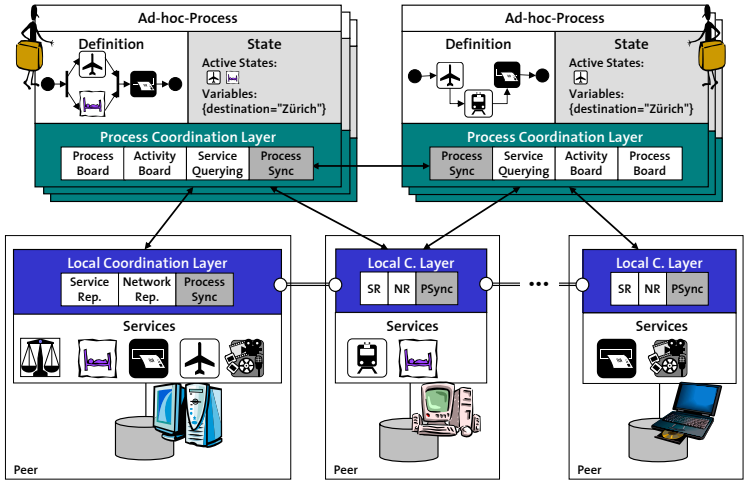


Abbildung 2: Aufbau des AMOR-Systems

3 User Management

Kleinrock [Kle95] stated "*... most of us are nomads, moving between office, home airplane [...] In doing so, we often find ourselves decoupled from our 'home base' computing and communication environment*". He also demands that in order to support nomadicity, computing should be independent of location, motion, computing platform, communication device, and communication bandwidth. This poses research questions regarding the

network layer, how to deal with decoupling or low bandwidth, or with the problems how to provide support if applications run on different devices (like discussed in [Koe03]).

Our work concentrates in the direction of having ubiquitous access to ad-hoc processes which are executed in a peer-to-peer manner without a centralized engine. In centralized approaches, this problem is reduced to implementing remote access to a server. For example, windows [WIN] allows to start applications on a server via terminal server. Then, the user can log out and log in from a different PC and can continue to work with the same session. Such an implementation is possible because the user location is unknown, but the PC knows on which server the login is required. This is different for peer-to-peer process execution, because users *and* processes move. This requires that either the processes discover the user or vice versa.

Our implementation follows the first approach. If a user is working on a device (or computer), this peer offers the service to communicate with the user. User interactions are modeled as *user interaction step*s of the ad-hoc processes. When they are executed, they send a query for a service 'Interaction with the user X'. If such a user and thereby the user's actual location is identified, the ad-hoc process migrates to the corresponding peer. Then, it opens a window for the user interaction. Certainly, such windows must be closed if the user logs out. As soon as he logs in again at another peer, the process transfers to the user interface respectively the process to that peer.

Figure 3 shows the local user management window implemented in the AMOR prototype. It is available on each peer, which users can use. On the left side, there are Login-Buttons to inform the user manager that a user works on this peer. Then, if a process that the user owns comes to a user interaction step, it communicates with the user via this peer. Also, a user which is logged in can start a new process (e.g. user Claudio could start the process evening.EveningProcess) which he is going to own.

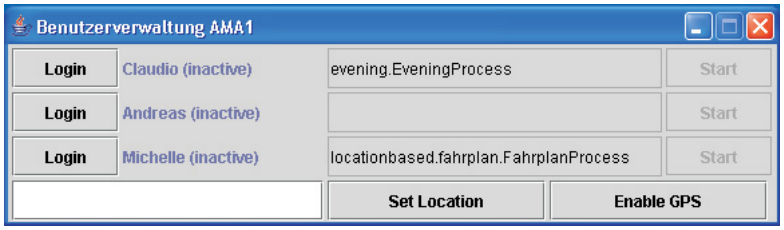


Abbildung 3: User Management Window for Peers

4 Location-Aware Services

[Wei91] stated „*If a computer merely knows what room it is in, it can adapt its behavior in significant ways without requiring even a hint of artificial intelligence*". This property is often known as location-awareness or context-awareness. Realizing this vision two aspects have to be addressed: First, the location has to be determined. This can mean to determine

the physical coordinates, which is sufficient for example in case of navigation systems. However, many applications require logical places like „I am at home“ or „I am near the main railway station“ which have to be derived from the physical location. Obviously, this is a hard problem which cannot be solved in a generic way.

Thus, the AMOR prototype focused on a sample of a location-aware timetable information service. The current logical location (modeled as a String) of a user using this service is received from the peer information the user is logged in. We support two ways to set this information (Figure 4): Firstly, the peer maintains static information which is entered by the user, for instance when the system is started. This is suitable for stationary peers like desktop PCs. Secondly, as mentioned above, a device can be used to determine the physical location which is then mapped to a logical place.

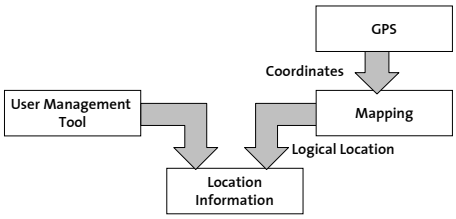


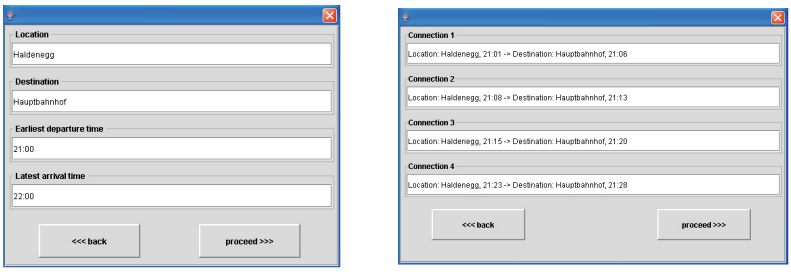
Abbildung 4: Architecture Location Management

In our prototype, we use one notebook for testing the automatic location ascertainment. The notebook is equipped with an Haicom HI-202E USB GPS mouse used together with the Chaeron GPS Library and IBM's Java Communication Library. This information – after it is mapped to a logical location – is used by an location-aware information system. It determines possible connections for public transportation. So our sample process of an evening planner gets the location as one input. Then, together with the information of the destination and the latest arrival time both coming from the decision support component (see next section) and the earliest departure time retrieved from the local peer's clock, it figures out possible connections. Figure 5(a) shows the input situation in which the locations and the times are set. Figure 5(b) presents the retrieved connections.

5 Dynamic Process Changes

Changing workflows were a hot topic in the second half of the 1990s [WAW98] and was mainly addressed with the term *adaptive workflows*. The proposed approaches did not only allow for alternative execution paths to deal with expected failure situations, but also provided support for failure cases not foreseen at the time the workflow has been modeled. It was already pointed out at that time that the possibility of changing workflows is important even in well-understood situations like medical information systems [Rei00] or workflows in public administrations [Sie98].

But if adaptability is important for the workflows in their 'simple' environment, then they



(a) Query (b) Result

Abbildung 5: Location-aware Service

are even more important for ad-hoc processes, which are used in situations which might appear only once or few times. Thus, they are useful also for not well-known situations and might be put together with thinking less about the problem than in case of workflows. So ad-hoc processes have to be able to support users in every-day situations, but especially there not everything can be foreseen.

Therefore, we introduced the idea of modifying processes with a simple decision support system. This decision support system provides specific support for the user in a sample scenario of planning an evening as part of the MAGIC project.[1] Figure 6 presents the modification of the AMOR architecture for this purpose. Whereas the collaboration between the process and the service providers remains unchanged, the process invokes now the decision support systems in conflict situations. Technically, we implemented this functionality by *expansion steps*.[2]

By this, we get an improved exception handling which can be supported automatically. Deiters et al. [DGJH+98] distinguishes four situations: basic build-ins, late modeling, post modeling, and model off-line execution. The first one is integral part of the AMOR system, which has a peer-to-peer transaction processing mechanisms as a basic feature [HS03a, HS03b]. The last one is only relevant for workflow systems which distinguish between a specification and instances. However, the important two other approaches are the two ways to use an expansion step in a process. It can be used either as the last step of a regular path because it is not known how to proceed (late modeling) or to cope with unexpected failures (post modeling). In the latter case, expansion steps are placed at points in the process where there is a change to find a new possibility to reach the overall goal.

Figure 7(a) provides an example for a process and a process expansion. The left column presents the initial process. The right one shows an expansion for one expansion step of

[1] This project is joint work together with the Artificial Intelligence Lab at the Swiss Federal Institute of Technology in Lausanne (EPFL).

[2] For our applications, it is possible to restrict the use of expansion steps to points in the workflow where no parallel path is executed simultaneously. Also, we do not have a definition of a workflow and instaces but just processes without this distinction. Thus, we can abstract from certain. Otherwise, crucial problems like the ones discussed in [RTB98, RRD03] would have to be taken into account.

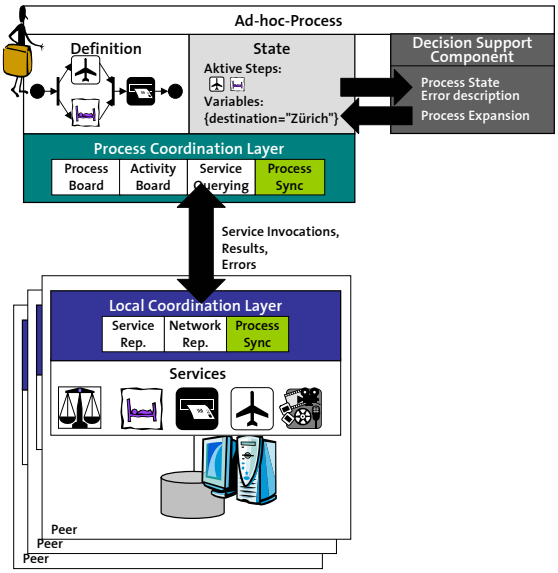


Abbildung 6: AMOR extended by decision support

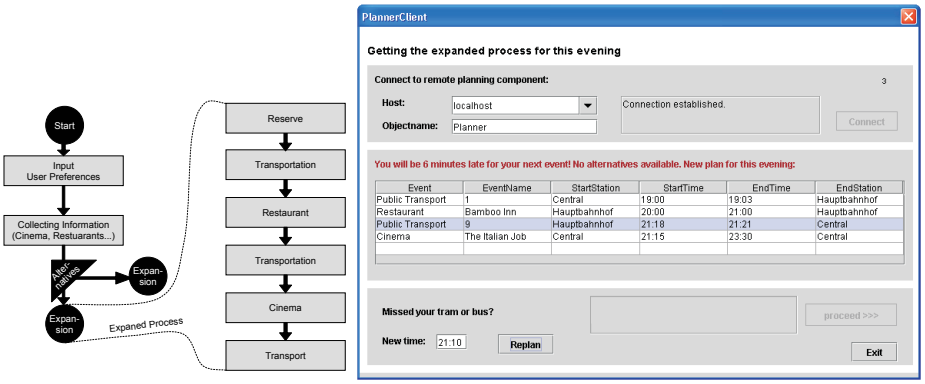
the initial process (for lack of space, we did not insert expansion steps in this column). The two expansion steps illustrate their two applications: The one as the provisional end of a path is placed there, because it cannot exactly planned what has to happen next. The user inserted his preferences and the system collected information. This information influences the further continuation of the process, e.g., depending on whether or not the user wants to go to a cinema or prefers to take a cab or to just hop on the next bus. Then, it can expand the process with new steps like the right column in Figure 7(a) resp. like the application example in Figure 7(b). The other expansion steps are in case e.g. the reservation fails. Then, the decision support system has to plan on the actual situation (user input, collected information, failure) how to proceed.

6 Summary

In this paper, we presented the results of two semester works, which extended the AMOR system by enhanced user support. AMOR, which a truly distributed system for peer-to-peer ad-hoc process execution, now supports the following three aspects:

1. Each peer provides a user interface that allows to model the existence or absence of a specific user. This extension allows an ad-hoc process to find and communicate with a user, although there is no global coordination component.

2. Based on the user management, a localization of the user is possible with the help



(a) Sample Process (b) Decision Support System Implementation

Abbildung 7: Plan Expansion

of the device the user is logged in. The location can be either hard-coded for a device (e.g. desktop PC) or dynamically derived with help of a GPS. The location information is used as input for a location-aware service.

3. The ad-hoc style of combining ad-hoc processes and the usage of them in unforeseen situations allows to dynamically change ad-hoc processes.

All these concepts have been demonstrated using an evening planning scenario. This scenario requires to adapt plans since it in the beginning of the process it is not clear what kind of preferences the user has for the evening. So, it is difficult to decide in advance what has to be booked and how many different events the user likes to visit.

The combination of user management and a location-aware scheduling information system enables to automatically plan for the user how to get from his current location, for example, to the cinema and to deliver the success of booking steps to whatever device the user is currently using wherever. Regarding these extensions, we have implemented a prototype for an ubiquitous ad-hoc process execution system with support for location-awareness and nomadic users.

Literatur

[Ack03] M. Ackermann. *Integration von ortsbasierten Diensten und Nomadic Computing in eine Ausführungsumgebung für Ad-hoc-Workflows*, Semester work, Database Research Group, Swiss Federal Institute of Technology (ETHZ), Zurich (to be submitted). 2003.

[DGJH+98] W. Deiters, Th. Goesmann, K. Just-Hahn, Th. Loeffeler, and R. Rollers. Support for exception handling through workflow management systems. In *Workshop 'Towards Adaptive Workflow Systems', Conference on Computer-Supported Cooperative Work, Seattle, WA*, 1998.

[GR93] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.

[HS01] K. Haller and H. Schuldt. Using Predicates for Specifying Targets of Migration and Messages in a Peer-to-Peer Mobile Agent Environment. In *5th Int. Conf. on Mobile Agents (MA), Atlanta, GA*, 2001.

[HS03a] K. Haller and H. Schuldt. Consistent Process Execution in Peer-to-Peer Information Systems. In *Proc. 15th Int. Conf. on Advanced Information Systems Engineering*, 2003.

[HS03b] K. Haller and H. Schuldt. Towards a Decentralized Implementation of Transaction Management. In *Workshop Grundlagen von Datenbanken, Tangermünde, Germany*, 2003.

[HST03] K. Haller, H. Schuldt, and C. Türker. Flexible Fehlerbehandlung für Ad-hoc-Workflows. In *Workshop Persistence, Scalability, Transactions - Database Mechanisms for Mobile Applications, Karlsruhe, Germany*, 2003.

[Kle95] L. Kleinrock. Nomadic computing (Keynote address). In *International Conference on Mobile Computing and Networking, Berkeley, CA*, 1995.

[Koe03] Th. Koenigsmann. Konzeption geräteübergreifender Anwendungen. In *Workshop Mobilität und Informationssysteme, Zürich, Switzerland* , 2003.

[Mun03] C. Munari. *Planexpansion und Fehlerbehandlung bei Ad-hoc-Prozessen: Konzeption und Implementierung anhand einer Beispielanwendung 'Eveningplaner'*, Semester work, Database Research Group, Swiss Federal Institute of Technology (ETHZ), Zurich (to be submitted). 2003.

[Rei00] M. Reichert. Prozessmanagement im Krankenhaus - Nutzen, Anforderungen und Visionen. *das Krankenhaus*, 11(92), 2000.

[RRD03] St. Rinderle, M. Reichert, and P. Dadam. Evaluation of Correctness Criteria for Dynamic Workflow Changes. In *Business Process Management, Eindhoven, The Netherlands*, 2003.

[RTB98] M. Reichert and P. Dadam Th. Bauer. Enterprise-Wide and Cross-Enterprise Workflow Management: Challenges and Research Issues for Adaptive Workflows. In *Workshop 'Towards Adaptive Workflow Systems', Conference on Computer-Supported Cooperative Work, Seattle, WA*, 1998.

[Sie98] R. Siebert. Anpassungsfähige Workflows zur Unterstuetzung unstrukturierter Vorgaenge. In *Workshop 'Towards Adaptive Workflow Systems', Conference on Computer-Supported Cooperative Work, Seattle, WA*, 1998.

[WAW98] Towards Adaptive Workflow Systems, 1998.

[Wei91] M. Weiser. The computer for the 21st Century. *Scientific American 265(3): 66-75*, 1991.

[WIN] Mircosoft Windows. `http://www.microsoft.com/windows/default.mspx`.