# A Workflow Architecture to Manage Mobile Collaborative Work

Henrique J. Domingos, J. Legatheaux Martins
Nuno M. Preguiça, Sérgio M. Duarte

{hj, jalm, nmp, smd}@di.fct.unl.pt

*Departamento de Informática*
*Faculdade de Ciências e Tecnologia*
*Universidade Nova de Lisboa*
*Tele. +351 1 295 4464, Fax. +351 1 2954658*
*http://www.di.fct.unl.pt*

**Abstract**

In this paper we propose a workflow−based system to manage and schedule group−oriented collaborative activities that scale beyond a conventional stationary organization. These activities must be seamlessly supported in a decentralized and distributed environment in which, stationary, as well as mobile workers, need to manage and execute individual *worklists* and related tasks. Individual tasks must be developed with certain levels of autonomy using local−processing capabilities. During the activities, mobile work performers are intermittently connected to a stationary infrastructure in which, partial contributions must be coordinated and adequately merged as tasks in the context of an organizational process.

## 1 Introduction

Recent research in CSCW systems (Computer−Supported Cooperative Work) shows that mobile people becomes interconnected in order to cooperate, by overcoming time and physical distances [Bergquist 99]. This type of active and quasi−permanent cooperation takes place everytime/everywhere and adopts synchronous as well as asynchronous modes of interaction and group−oriented resource sharing.

The cooperation activities of mobile workgroups are difficult to manage with traditional CSCW systems and are very hard to coordinate with conventional workflow management systems (WfMS).

To address the problem of providing a coordination and cooperation environment, in which mobile and stationary workgroup interactions can be supported, some new requirements must be considered. Those requirements are closely related with the following aspects:

- Support for flexibility, promoting collaboration−aware resource sharing but preserving adequate levels of autonomy of individual mobile workers;

- Support for resource sharing facilities, participation and awareness control about the tasks performed by work performers, in a group−oriented work perspective;
- Coordination support by means of an adequate workflow management model specially tailored for flexible task assignment and scheduling for mobile and intermittent connection environments.

In this paper we further discuss these requirements and solutions fulfilling them. In the section 2 we introduce WfMS, discussing some limitations when such systems are used to manage collaborative and mobile work teams. Section 3 summarizes an empirical study concerned with a typical workday of mobile work teams acting in an electricity provider company. The section 4 presents our proposal, while the Section 5 is dedicated to related work. Finally, the section 6 concludes the paper.

## 2. WfMS: background and limitations

### Background

A WFMS is a proactive software system that manages a flow of work tasks performed by different participants, following defined procedures to achieve an objective. It manages users by controlling their participation with different roles together with appropriate data−management resources, which may be directly accessible trough the system as well as indirectly by means of external applications. The objective is achieved by setting deadlines, task synchronization conditions and by passing task's data from one participant to another in correct sequences, ensuring that all fulfil their required contribution and taking default actions when necessary. Basically, a WfMS can be seen as the support for procedural automation of work processes, by managing, monitoring and auditing the sequence of tasks.

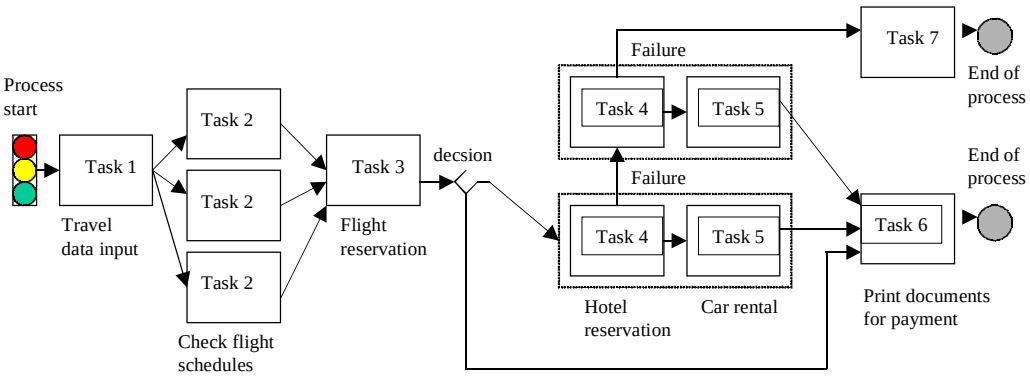


Figure 1: A visual representation of a workflow

Figure 1 shows a conventional representation of a typical workflow process: a business travel trip reservation.

In a conventional WfMS, there are two crucial and separate moments. First, a definition and specification phase takes place. This is followed by an execution or activation phase − carrying out the pre−defined process. Gradually, this basic approach is evolving to new models in which there are two extensions to the above approach. The first one is to use certain forms of organizational knowledge to adjust particular tasks and to cater for

unexpected inputs or results in each task. The second extension is the object of recent research work and aims at bringing the definition phase itself adaptive. The goal is to allow dynamic tailorability of the task definition, during its execution, in order to satisfy unexpected goals, conflicts and divergences of the performers.

The above extensions are difficult to achieve with the conventional WfMS architectures based on the reference model proposed by the Workflow Management Coalition [WfMC 99], which is represented in the figure 2.

The relevant components in such architectures are: the process definition and specification tool, the administration, monitoring or auditing tool (or tools), client applications managing user's worklists and the enactment or runtime support environment. Usually, external or independent specific applications can also be involved and can be automatically activated to support or to assist users in performing their tasks.

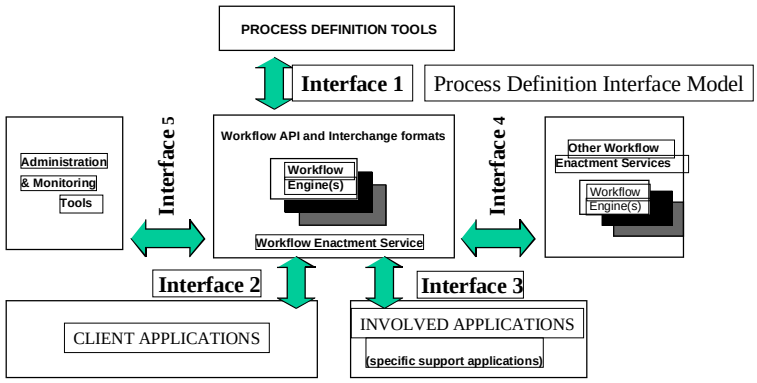


Figure 2: The WfMC reference model

The design and specification tool(s) are administration process definition tool(s), mainly used to specify organizational (also called business) processes. A process is regarded as a workflow composed by a sequence of defined tasks. The process definition requires a specification language (usually a scripting language). There is no commonly accepted basis for language standards. WfM specification languages are in general used to describe acyclic graphs of tasks−definitions, colored and enriched with pre−conditions and flow−control, controlling state−transitions. Most WfMS provide graph visualization tools with nodes representing tasks and arcs representing data−flows and flow−control between those tasks. The execution of a workflow can be regarded as the concurrent execution of a state−machine, where the current state of each task execution (work performer) is associated with the task being currently performed.

Auditing and logging tools are management tools aimed at monitoring and tracking the progress of the executing workflows.

User's worklist management tools are used to distribute work assignments, allowing users to enter state transitions and other relevant data for the control and management of each organizational process.

The enactment system is the runtime execution system support for workflow instances and task activation. This runtime system controls and synchronizes each task in the workflow and assigns them to specific organizational resources and participation roles. These resources can be humans, applications or even other workflow systems. As traditional WfMS are

sophisticated process specification/modeling/simulation software pieces and activation/execution runtime environments, the more representative ones are generally rather closed. Often, the same vendor usually supplies all the components of the WfMS.

To avoid this excessive centralization, the WfMC proposed recently a standard reference model (sketched in the figure 2), defining terminology and interfaces between different WfM conceptual components.

**WfMS and Mobile Cooperative Work**

The effectiveness of workflow processing is central to the success of many organizations with more effective workflow systems leading to improved corporate competitiveness. The main advantages are:

- to the work performers – because each process is well defined or documented, data is supplied automatically and work−package status is permanently visible;
- to the system−designers and maintainers – because the development of customized systems can be simplified;
- to the work managers – because a WfMS allows for process measurements and analysis, adherence to preferred processes by simulating specific policies, more efficient resource utilization and support for continuous process improvements;
- for the customers (or final users) – allowing shorter task−flow times, standard procedures and status control.

In despite of these advantages, conventional technology only supports pre−planned work, which calls for standard tasks in the belief that this warrants good quality outcomes by following pre−specified, well−known and pre−defined rules. Business and organizational processes, however, are now becoming partially volatile following complex work practices requiring that certain WfM levels no longer rely on pre−specified or well−known actions. Collaboration and mobility of task−participants are of course among the key factors driving this new reality.

In fact, the conventional approach for WfMS (as proposed by WFCM) is not well suited to fulfil the requirements of collaborative−oriented tasks performed by users in a decentralized computing environment integrating mobile computers. WfM technology implements rigid hierarchic models of task coordination, over which, the workflow system administration has, centrally, a total or significant authority. This approach has obviously organizational difficulties as well as a deep impact on the architectural models and system design options[1].

In mobile as well as collaborative tasks, this approach becomes useless and only implements a fiction [Ellis 98]. In mobile collaborative scenarios, pre−defined processes are too restrictive and there is high probability of conflicts/divergences as well as communication failures, involuntary disconnection and more problems to deal with high availability and reliability in resource sharing. Strict or rigid concurrency control mechanisms based on strong semantics are certainly an enormous limitation.

[1] In despite that the organizational difficulties and technological issues can be closely−related, we are concerned with the design options to introduce more flexibility and decentralization in the WfMS, at system support level.

Some limitations of traditional WfMS architectures are due to the fact that the enactment system is implemented in a central server. This server contains a repository of workflow definitions, assets, resources and drives process−execution and synchronization, task−binding, role assignments and task distribution, monitoring, auditing and management.

A very important limitation is concerned with the fact that the WfM server is supposed to host and coordinate users worklists on behalf of each participant, which is very problematic in the case of mobile workers.

WfM architectures for mobile and collaborative tasks must start by accepting the asynchronous nature of mobile and collaborative interactions, which take place on different−time / different−place conditions related with complementary periods of connection/disconnection. A flexible approach for the reconciliation between autonomous work/local processing (during partial disconnected work) and consistent workgroup management vision and task's synchronization (optimizing periods of connected work) is a key−criterion in the design of new WfMS architectures for widespread collaboration and coordination environments. Furthermore, in cooperative work scenarios, the existence of incomplete task definitions after their activation is usual. Evolving task definitions should be completed with information only obtainable on physical locations. This relates with capturing *in−loco* information concerned with diagnosing, foreseeing and gradual result reporting, in intermittent cycles of connected and disconnected work.

Complementarily to the task's synchronization conditions and state−transitions, which are in general strong−semantics operations, the system requires active notification mechanisms, disseminating critical modifications on work pre−conditions and/or awareness control about the user's operations. Notification and group−awareness control needs are not covered in conventional workflow technology, in which each participant has a partial and isolated view of her/his own contribution. Awareness control facilities extending the work representation to accommodate smoothly possible task interdependencies in a flexible mediating system between a reference work representation and the dynamic user's actions, seems to be more appropriated.

Case−Study: observing mobile work teams in a energy provider

To illustrate a possible scenario for the need of new WfMS architectures we introduce, briefly, an empirical study and some observations from the electrical industry.

Most electrical companies are organized in a central office and several (regional) service station offices. Each service station is responsible for a geographical region of the electrical network and is the base of several work teams. There are three relevant roles in each station: administration staff, planners and electricians. Electricians act in mobile work teams in the day−to−day technical operations, repairing and developing the network. A planner is a responsible for planning and supervising those work activities performed by the electricians and other external personnel. Each planner is responsible for about 5 teams of electricians.

In our empirical observation, we distinguished different phases in a typical workday: data and knowledge sharing, work−assignment, task's−execution, execution control and work reporting. These five phases occur in cycles of one day, with minimum variances in the time duration associated with each phase (as summarized in the figure 3).

During the execution phase different actions can occur: simple notifications, diagnosing annotations, foreseeing and work−rearrangements. Each phase distinguishes different work contexts in which, group−oriented collaboration and coordination have different levels of relevance and emphasis, as represented in the figure 3. A more detailed description of this

environment and work characterization as well as the way the mobile teams are coordinated is presented in [Domingos 99]. In the remaining paragraphs we introduce a general purpose WfMS aimed at the coordination of mobile work teams in environments similar to the one presented above. We believe that this type of scenario is frequent in all industries with mobile work teams performing repair work or connecting customers to different utility infrastructures.

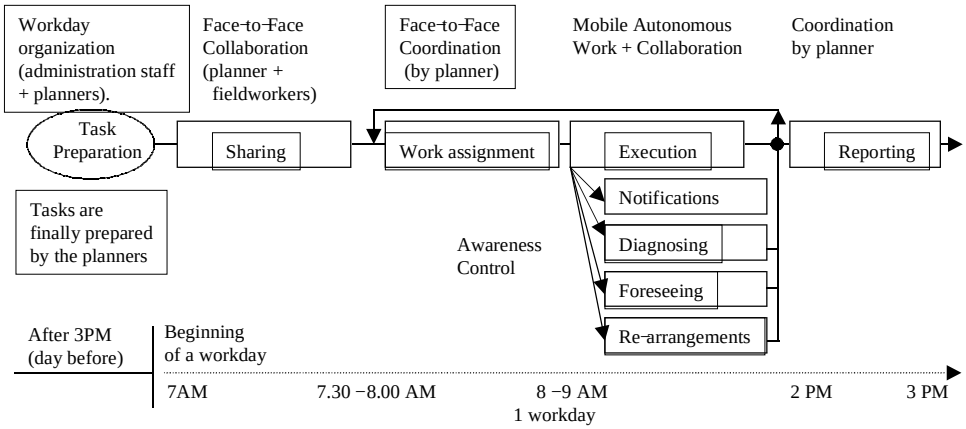


Figure 3: Summary of a typical workday

Proposed workflow architecture

4.1. Concepts and abstractions

A Workflow Management System (WfMS) helps with the management of the different interventions, reparations, installations or other work activities that take place in a company. We propose a WfMS architecture supporting an hierarchy of decentralized organizational workflows with different levels of specialization and work−control granularity [Domingos 98]. The hierarchy is aimed at supporting adaptability by not starting by over specifying the tasks. To this end, it starts by *a meta−level organizational workflow* where tasks are themselves (stub) workflows and, recursively, can be refined up to *task−instantiation workflows*. In our case study, such workflows are further refined up to the level of *task−assignment workflows*, managed at the level of regional service station offices, where local planners take care of them.

Task−assignment workflows, or simply workflows from now on, are, in general, directly associated with field work and their execution lasts from one day to one week. They can manage repair work, small projects (as for example the ones associated with some reconfigurations or the expansion of the electricity network in our scenario), interventions in customer premises (as for example the ones needed to connect new buildings), etc. A workflow managed by planner relates all the tasks to be performed by teams under his coordination and responsibility. There are three main roles to be considered: administration staff (people that access, centrally, to the current state of the workflow execution), planners which are the owners of workflows and teams, which are the final work performers[2]. We

[2] Note that we use the term “team” to refer the work performers, to emphasize the collaborative nature of the work performed locally in different places. Obviously, the system supports tasks to be performed by only one mobile worker (as a singular work team).

define a *worklist* as a set of tasks from the same workflow assigned to the same team. A set of worklists of a team is a workwallet. Workwallets are coordinated autonomously by the teams.

### 4.2. System Architecture

Administrative staff and planners have access to the WfMS which is, at first sight, similar to a traditional, centralized, client / server system. The *core* of this system is supported by a conventional stationary computer infrastructure based on a fixed high performance computer network interconnecting the different premises of the company. Due to the fact that planners often need to visit the work teams in the field, they have mobile computers. While at stations, these computers are wired to the facilities network as stationary computers. While at field, they are intermittently connected with the help of a wireless connection, of lower quality. Administrative staff only uses stationary computers. Some of the work teams are also equipped with mobile computers, not only to access the WfMS, but also to access the company's databases of technical information while on field (repair manuals, parts manuals, electrical schemes and so on). Other teams only have access to small hand−held computers that can also be intermittently connected via wireless paths or serial wires of lower performance.

The core system supports the common operations of a WfMS and stores the information and state of executions associated with each workflow. Relevant information about each workflow comprises its owner (a planner), the definition and scheduling of its tasks, the current execution state of each task, etc. Some of the information has append−only characteristics allowing concurrent updates performed by different users to be easily merged – for example, each task has an associated annotation object where planners and workers log their comments, diagnoses and other information. One important feature in our system is the task assignment process by which teams become responsible for the execution of tasks and that can be performed in a closed or open way, as it will be detailed later.

The status of workflows, planners and teams can be partially cached in the mobile computers allowing planners and teams to access this information offline. While planners usually cache all the information associated with the workflows and teams they are responsible for, teams only store the information about their assigned tasks (and possibly the ones they are competent to perform and that are not assigned to any other team). To help teams to coordinate their activities and schedule their assigned tasks – which can belong to different workflows – they have a local autonomous application to manage their *workwallets*.

Our system relies on the premise that the core system always has the *official* information about all workflows and that updates are only valid when applied and stable in the core system. The concurrency control policy is based on a revocable promise locking mechanism that allows disconnected users to update the information about workflows if they hold the appropriate promise locks (note that append−only objects may always be updated without any locks). These updates are initially applied to the cache in the mobile computers and later propagated to the core where they are reintegrated in the official state. In a later section we detail the outlined mechanism and explain how do we handle the rare situations where locks have been revoked.

#### Caches, event−channels and awareness control

Associated with each workflow there is a *N−to−N event−channel*. Events related with the state−transitions of the workflow and its different tasks are propagated throughout this channel. Associated with each planner there is also another N−to−N event−channel. Planners and the teams they coordinate are allowed to send and receive events to or from these channels. All these events are logged by the core system in a persistent way. Both channels can also be used to propagate awareness events not directly related to object updates.

To lower the support requirements to a bare minimum, we anticipate that these event−channels can be supported by multicast communications allowing N−to−N communication in a not ordered and not reliable way, like a simple SMS channel or an e−mail list. As so, persistency is only warranted on events that get to the core. This is a likely situation not completely warranted by the system. However, all events sent by the core are persistent and easily ordered. We anticipate that events sent from a mobile computer, reporting a local object update, are again multicasted when the core sends a similar event, reporting that the same update got to the core and becomes stable. Such duplicates are also easily detected and eliminated.

Mobile devices can cache workflows, teams status, tasks, worklists and workwallets. Naturally, a planner will have a bigger cache then a team. In this last case, in the limit, the mobile device may only be capable of caching the tasks the team is involved on. Workflow and task assignment provides simple hints about which objects should be cached in teams or planners devices.

The status of the different workflows managed by the system is the one that the core system reports. It can only be updated by applying update transactions to the core. Due to the properties of the event channels above, an update of an object performed while its supporting computer is disconnected, can be logged by the core system and even be seen by other teams, but be later aborted due to the fact that it violates concurrency control requirements when it finally gets safely propagated to the core.

Side by side with cached objects, the mobile device must also record the updates received from the workflows event−channels, as well as the updates performed locally. Whenever possible, object caches are maintained in two versions. One version is the last officially consistent state received from the core system. The other version is a tentative local one.

Local updates performed by the user are applied to the tentative version[3], propagated by the event−channel associated with the tasks workflow and locally logged for later safe propagation to the core. Remote updates, received from remote teams through the workflows event−channel, are also applied to the tentative version.

Whenever the mobile can connect with enough quality of service to transmit safely all local updates, and to receive new consistent copies of the objects cached, the two copies can become similar. Updates on objects not locally cached are possibly recorded without further computing.

Due to the concurrency control policy of the system (see below), local updates are likely only applied on objects currently locked by the mobile device owners and, except in rare situations, no remote update will be received on these same objects.

[3] The management of tentative versions is optional. Very small devices can only record the stable versions, the log of local updates to propagate, if any, and a round−robin buffer of the last events received.

### Task assignment and locking

Unassigned tasks are assigned by planners to teams in a closed or in an open way. With the closed model, work teams can access the system to query about task assignment and task definition because planners have already performed and propagated the updates about their decisions related with task assignment. This can take place during the daily early morning meetings, at stations, where good connectivity conditions are available. Anyway, the technology also allows for the teams to receive the assignments remotely by downloading the worklists assigned to them.

When a planner assigns a task to a team, the task gets locked by the team. If the team loads a copy of the task status in its mobile computer, its cache is always updated on this task because of the lock. However, these locks are *revocable promise locks* and must be re−stated periodically by the team. While the team keeps the lock, their members can be assured of the success of their updates, even later propagated. If the team looses the lock, nothing can be assured and the task becomes *tentatively assigned* at the core and locked by the planner. The planner can then re−assign the task to the same team, or to another team, or can re−assign it the open way. The period a team has to re−state a lock is tunable and is related with the time frame needed to complete the task, what is usually closely controlled by the WfMS. Planners are allowed to reassign or unassign a task at any moment. If this event takes place, it is propagated immediately to the workflow event−channel and the associated conceded locks are revoked. Also, the planner of the workflow implicitly locks all tasks not locked.

Updates performed by disconnected teams may be considered invalid if when they are propagated to the core system the team no longer holds the appropriate locks. In this situation, the updates performed are logged and may be used by the planner to reintegrate the relevant information – note that updates performed to the append−only objects are never invalid because no locks are required to update them.

The open way of assigning tasks to teams takes place by publishing task availability via the planner event−channel. Subscribers to this channel can be the teams, being in the field, or external workers that act as sub−contractors. The economic or personnel management reasons that make this scenario realistic are behind this paper. Anyway, it is easy to imagine that people in the field can be re−assigned in an open way due to unexpected urgent situations. A classical rigid WfMS can not deal with this situation, which is often managed outside the system. As work rescheduling and re−planing, or work out−sourcing, are more and more frequent in real life, the WfMS must allow for its management.

As open task assignment is a coordination process driven by participant teams and the planner and not only by a coordinator (i.e., the planner), the system must accommodate several forms of elections. The first form of open task assignment chooses the team that gets assigned to the task as the first one that makes a transaction in the core system to get the task. The second form chooses by mutual agreement after some form of field or remote coordination[4]. In the last two scenarios, the planner has the final word because up to his decision it owns the tasks lock. When the task is finally assigned to a team, the system starts a lock treatment as performed in the closed assignment process.

### Using the system during the mobile fieldwork

[4] Other scenarios, like voting on task assignment or different forms provided by other group−decision tools can be defined.

Teams involved with an workflow always have a vision of tasks being assigned or open. In the course of the work, they are aware of the tasks that are assigned to them, and can also be aware about the execution state of the tasks being performed by others. If the tasks are their responsibility, they own a lock on them and their status, even when cached, is probably accurate. They can also see new tasks being generated or communicated and can, in a cooperative way, decide to participate in the process of performing some of them.

When a team is responsible for a task depending on the execution of another task, it may use the information propagated in the workflow channel to be aware of the conclusion of that task. Note that teams may use all notifications and not only the ones propagated by the core workflow, to make locally their decisions. In fact, if the team responsible by the ending task has sent the event, that team had a lock on that task, what was probably known to the receiving team. Thus, the receiving team can proceed, because later or sooner the update announced by the event will be safely propagated to the core system.

When the different events arrive to each team, they allow the rescheduling of local workwallets by the applications managing autonomously the team's workload. Note that the event−channels above complement the natural usage of informal awareness channels (mobile phones, SMS, etc,). In our case study, a persistent unicast channel between each team and each planner supports the supplying of persistent foreseeing notifications. Diagnosing is based on awareness control information disseminated within the workflow channel, which is a persistent N−to−N channel. The planner channel is used to announce open tasks upon dynamic rearrangements, during the workday. Simple awareness annotations are propagated to the core workflow and also disseminated.

Related work and Discussion

Several projects are exploiting the support of mobility in the context of WfMS. Different systems vary in the way they deal with task assignment models, pessimistic/optimistic strategies for task allocation and locking, semantics of disconnected operations, control execution and scale support. The recurring discussion about the functionality placed on disconnected clients is also a key−criterion: the replication of part of the functionality is interesting in terms of high−availability and in preserving adequate levels of autonomy for task performers. However, this complicates the system design in a possible significant way.

The approach of the Exotica system [Alonso 97] is based on a decentralized architecture for mobile and disconnected task execution based on a pessimistic approach: prior to a disconnection, each mobile client locks the tasks planned for local execution and hoards relative to that tasks data. The implementation is based on the concept of worklists that conceptually are sets of workitems[5] to be executed by single work performers. Everytime a workitem is eligible for execution, the fact is (centrally and reliably) broadcasted to the worklist of all the users associated with the same task. When a user selects a workitem from his worklist, this task is deleted from all other worklists, and is associated exclusively to that user. This forces that each user locks tasks selected before a disconnection from the workflow server. Then, the relevant data for the task is cached and stored locally. During a disconnection, clients work only on tasks previously locked, and when the task is finished, state−transitions are stored persistently in the client stable memory to be reintegrated upon a reconnection.

[Klingmann 97] also proposes a system (Transcoop) based on exploiting the operation semantics to reduce the need of pessimistic locking on long−transactions, allowing the possibility of merging operations executed at disconnected sites using an history merging

[5] These workitems correspond to workflow tasks in our proposed architecture.

mechanism. The authors report difficulties in adopting this mechanism in a generic way, particularly in the case of applications that may involve thousand of clients.

A pessimistic approach (like in Exotica and Transcoop) is often justified by the large number of conflicts, causing high abortion rates in optimistic scenarios, due to the high degree of data sharing needed for cooperative work. While using a mix of pessimistic and optimistic approaches, we exploit the semantics of coordination attributes and the role of planners, which allow a hierarchic structuring of work responsibilities. This avoids, in a sound way, the above drawback, allowing at the same time an high degree of caching and data sharing improving flexibility and high−availability on task management.

We argue that replication and caching are the main ingredients to cope with large−scale settings and resource collaborative sharing. These two techniques have practical problems in scenarios where an high degree of coordination is needed among large groups of users and tasks objects. In our opinion, however, our hierarchical and recursive organization of workflows at different coordination levels are a sound way of managing teams of mobile workers that avoids the horizontal growth in the dimension of the groups that a flat structure would introduce. Warranties associated with locked tasks assure that for critical tasks there are no divergences in the state of distributed executions. The role of the planner, in our case study, shows the pragmatic of our solution in many real situations of mobile work.

Other architectures are based on mobile agents, where the idea is to avoid a centralized workflow server. A local agent executes autonomously by encapsulating private data of the workflow, a set of pre−conditions controlling the flow between the tasks of the agent computation and a log of operations locally executed. Each mobile agent is submitted to a processing entity and roams among processing nodes to achieve its objective. They can support operation among partially disconnected processing nodes. The execution of local agents supposes strong warranties based on transactional semantics (ex., [Barbarà 97]). Prototypes based on these architectures are however very simple applications [Hawryszkiew98]. Scale, reliability, awareness control and agent's execution control, are for example, open problems. In despite of the interest of these architectures in possible scenarios (different than our case studied) many problems remain as open design issues: security, access−control and global control execution are some examples.

RainMan [Santanu 98] uses a task−assignment model based on an event subscribing policy on work dissemination channels, by means of call−backing remote calls when users bind to the central workflow. The authors refer that this model has several advantages to meet the scale and mobility requirements. Other main design issues, dealt in different ways by more recent research projects, are also related with scalability concerns, dynamic task reconfiguration and adaptive work assignment. Projects like ([Alonso 99], [Han 99] or [Heinl99]) propose reliable event notification services as alternatives to support scalability and mobility concerns, as well as allowing awareness control, collaboration and decentralized execution of adaptive WfMS. We also use event channels for the same objectives but avoid using reliable channels due to the problems that a decentralized and peer−oriented implementation of reliability brings. Lower requirements, we hope, will allow easier implementations and a richer usage of emergent wireless communication facilities.

Conclusions

In this paper we proposed a flexible architecture to coordinate mobile collaborative work teams. Our motivation and contribution is concerned with the limitations of classic WfMS in managing collaborative activities and mobile work performing.

The system proposes a simple way of synchronization of the different mobile work teams

based on the observation that work planners generally centrally coordinate these teams. We allow a flexible mix of optimistic and pessimistic strategies minimizing update conflicts and transaction abortion while allowing almost lock free updates of workflow objects cached by mobile computers. Event notification services are used to promote collaboration, awareness− control, and state convergence and lock status propagation. All these facilities are available without penalty to a simple semantics model very similar to the one we find in traditional centralized systems. We believe that the architecture proposed is a possible generic solution that can be applied (with minimum adaptations) for all industries with mobile work teams performing repair work or connecting customers to a utility infrastructure.

## References


[Alonso 96] G. Alonso, G. Gunthor, M. Kanath, D. Agrawal, A. Abbadi, G. Moham, “The Exotica/FMDC: Workflow Management System for Mobile and Disconnected Clients”, “Distributed and Parallel Databased, Vol.4, N.27, 1996

[Bergquist 99] Jens Bergquist, P. Dahlberg, F. Ljungberg, "Moving out of the meeting room: Exploring support for mobile work", in proc. of the international European Conference on Computer Supported Cooperative Work − ECSCW 99, Copenhagen, September 1999

[Domingos 98] Henrique J. Domingos, J. Legatheaux Martins, Nuno M. Preguiça, Sérgio M. Duarte, "Coordination and Awareness Support for Lareg Scale Collaborative Work Sessions", in Proc. of the 4th CRIWG International Workshop on Groupware Systems, Brasil, September 1998

[Ellis 94] R. Ellis, "A Conceptual Model for Groupware", in Proc. of the ACM International Conference on CSCW, 1994

[Hagen 99] C. Hagen, G. Alonso, "Beyound the Black Box: An Event−Based Inter−Process Communication in Process Support Systems", in ICDCS 99 − International Conference on Distributed Computing Systems", Austin Texas, June 1999

[Han 99] Y. Han, A. Shet, (METEOR Project Team − University of Georgia) , "On Adaptive Workflow Modeling", in Proc. of the 4th International Conference on Information Systems Analysis and SynThesis, pp. 108−116, Orlando Florida, 1999

[Hawryszk 98] I. Hawryszkiewycz, J. Debenham, "A Workflow System Based on Agents", in Prc. of the DEXA 98, 9th International Conference on Database and Expert Systems Applications, pp. 135−144, September 1998

[Heinl 99] Heinl, P.; Horn, S.; Jablonksi, S.; Neeb, J.; Stein, K.; Teschke, M.," A Comprehensive Approach to Flexibility in Workflow Management Systems", (Mobile Workflow Project Team, University of Erlangen") in Proceedings of the ACM 1999 Conference on Work Activities Coordination and Collaboration (WACC’99), San Francisco, 1999

[Santanu 97] "RainMan: A Workflow System for the Internet", in Proc. of the Usenix Symposium on Internet Technologies and Systems, pp. 159−170, December 1997

[WfMC 99]. Workflow Management Coalition (WfMC documentation) − http://www.wfmc.org

[Domingos 99] Henrique J. Domingos, J. Legatheaux Martins, Nuno M. Preguiça, Sérgio M. Duarte, “A Workflow Architecture for Mobile Collaborative Work: a Case−Study and its Requirements”, Technical Report DI−UNL−02−99, DI−FCT−UNL.