Workflow Applications to Research Agenda: Scalable and Dynamic Work Coordination and Collaboration Systems

Amit Sheth and Krys J. Kochut

Large Scale Distributed Information Systems Lab, Computer Science Department, The University of Georgia Athens, GA 30602-7404

1. Introduction

A workflow is an activity involving the coordinated execution of multiple tasks performed by different processing entities [KS95]. These tasks could be manual, or automated, either created specifically for the purpose of the workflow application being developed, or possibly already existing as legacy programs. A workflow process is an automated organizational process involving both human (manual) and automated tasks.

Workflow management is the automated coordination, control and communication of work as is required to satisfy workflow processes [SGJ+96]. A Workflow Management System (WFMS) is a set of tools providing support for the necessary services of workflow creation (which includes process definition), workflow enactment, and administration and monitoring of workflow processes [Hol94]. The developer of a workflow application relies on tools for the specification of the workflow process and the data it manipulates. The specification tools cooperate closely with the workflow repository service, which stores workflow definitions. The workflow process is based on a formalized workflow model that is used to capture data and control-flow between workflow tasks.

The workflow enactment service (including a workflow manager and the workflow runtime system) consists of execution-time components that provide the execution environment for the workflow process. A workflow runtime system is responsible for enforcing inter-task dependencies, task scheduling, workflow data management, and for ensuring a reliable execution environment. Administrative and monitoring tools are used for management of user and work group roles, defining policies (e.g., security, authentication), audit management, process monitoring, tracking, and reporting of data generated during workflow enactment.

Workflow technology has matured to some extent, and current products are able to support a range of applications. Nevertheless a majority of the workflow products are directed towards supporting ad-hoc or administrative workflows that serve as office automation types of applications, and primarily involve human tasks supported through forms-based interface. Support for more complex, production workflows is limited to more repetitive processes.

Many additional limitations remain, especially for supporting more demanding applications, more dynamic environments and for better support for human involvement in organizational activities. In this paper, we focus on two issues. The first issue relates to the challenges that could be addressed by evolving the current workflow technology. Two of the challenges to which we focus our attention in this paper are: (a) support for scalability, and (b) support for adaptable and dynamic workflows. Various other research challenges have also been recognized, and for some of them the research is further along.

The second issue we address in this paper is somewhat less well defined and longer term. It mainly relates to supporting a broader aspect of organizational activities. To support these, the workflow technology, owing to its emphasis on coordination, is only a part of the solution. Following the recent NSF workshop on Workflow and Process Automation in Information Systems[She96], the report by a multidisciplinary group of researchers noted [SGJ+96]:

> "Work Activity Coordination involves such multidisciplinary research and goes beyond the current thinking in contemporary workflow management and Business Process Reengineering (BPR). In particular, instead of perceiving problems in prototypical terms such as the information factory, white-collar work and bureaucracy, we believe that this limited point of view can be explained by a lack of synergy between organizational science, methodologies, and computer science. Multidisciplinary research projects, based on mutual respect and willingness to learn from another discipline, can help to create a thriving research community that builds upon the strengths of different disciplines, such as distributed systems, database management, software process management, software engineering, organizational sciences, and others."

Numerous multidisciplinary research issues, especially those relating to effective support for and providing satisfaction to humans as they participate in the organizational activities remain. At a technological level, two rapidly maturing technologies present special opportunities as they are complimentary to coordination focused workflow technology. One is the collaboration (and group work) technology, including video-conferencing and shared spaces, where the focus has been on synchronous communications and/or support for human intensive and ill defined (or ad-hoc) activities. The second is information management, including integrated access to the organization's (heterogeneous media) data or information assets. In this paper, we will only give some preliminary thoughts to the appealing research agenda of increasing integration of coordination (workflow), collaboration and information management technologies to support complex and fluid work activities in real-world organizations. At the technological level, Web, distributed object management (specifically CORBA), and Java allow us to interface and integrate disparate technologies with increasing ease.

The rest of this chapter is organized as follows. Section 2 provides a brief overview of the current workflow technology. In Section 3, we discuss short application scenarios that provide us motivation for research in scalability, adaptive and dynamic workflows, and combined support for coordination and collaboration. Section 4 contains an overview how our METEOR$_2$ workflow management system addresses the issues of scalability. Finally, Section 5. presents our early work on architecture and design of a Workflow Coordination and Collaboration System to address the other two challenges of adaptable/dynamic workflows and integration of coordination and collaboration technologies. Conclusions are provided in Section 6.

2. Agenda for Current and Future Research

Workflow Management Systems today are being used to re-engineer, streamline, automate, and track organizational processes [JAD+94, GHS95, Fis95]. There has been a growing acceptance of workflow technology in numerous application domains such as telecommunications, software engineering, manufacturing, production, finance and banking, laboratory sciences, health care, shipping, and office automation. The current state-of-the-art in WFMSs is dictated by the commercial market [AS96]. Several hundred products that provide support for workflow management (in various degree) exist in the market today [Sil95, Fis95, GHS95, SJ96, AAAM97]. These systems are primarily focused toward providing automation within the office environment with emphasis on coordinating human activities, and facilitating document routing, imaging, and reporting.

In the last few years, pervasive network connectivity, catalyzed by the explosive growth of the Internet has changed our computational landscape. Centralized, homogeneous, and desktop-oriented technologies have given way to more distributed, heterogeneous, and network-centric ones. This has raised challenging requirements for workflow technology in terms of being required to support large scale multi-system applications, involving both humans and legacy systems, in heterogeneous, autonomous and distributed (HAD) environments. Emerging and maturing infrastructure technologies for communication and distributed object management (e.g., CORBA/IIOP, DCOM/ActiveX, Java, Notes, and Web) have addressed many of these challenges, and are making it feasible to develop standards-based large scale, distributed application systems.

As a commercial technology, workflow technology has undoubtedly experienced significant success. This has lead to the desire for further research and development in the area for two directions. First, the requirements placed on some of these applications have tested the limits of the current workflow technology, highlighting the need for further research and development. Second, researchers and practitioners have identified new applications that could con-

ceptually use this technology, but current workflow products, either do not address these emerging requirements, or do so very selectively.

Some of the apparent weaknesses that need to be addressed from the perspectives of database and distributed systems community researchers active in the workflow management area include limited support for heterogeneous and distributed computing infrastructures, lack of a clear theoretical basis, undefined correctness criteria, limited support for synchronization of concurrent workflows, lack of interoperability, minimal scalability and availability, and lack of reliability in the presence of failures and exceptions [BDSS93, JNRS93, GHS95, AS96, KR96, LSV96, WS96, WS97, AAAM97]. The issues of error handling and recovery [JNRS93, AKA+94, Ley95, EL96, WS97, RSW97, Wor97], and of role of transactions and transactional workflows [SR93, GHS95, TV95, KR96, LSV96, WS97] have however recently seen some attention, but will not be discussed further in this paper.

Researchers and practitioners in other relevant communities have correspondingly identified many other items that call for further research. The report for the multidisciplinary team from the NSF workshop identified the active research areas to include [SGJ+96]: Definition and Modeling; Representation, Language, and Meta-Modeling; Analysis, Testing, Verification and Evaluation; Simulation; Prototyping, Walk-through, and Performance Support; Administration, Staffing and Scheduling; Interoperation and Integration; Target Support Environment Generation; Monitoring and Measurement; Visualization; Enactment History Capture and Replay; Fault Detection, Error Handling or Repair; and Evolution, Continuous Improvement and Model Management.

3. Application Driven Motivation and Requirements

In this paper, we focus our attention to three of the many interesting challenges facing the workflow technology and requiring additional research. Where possible, we use application examples that have provided motivation and helped us understand some of these requirements. Discussions of workflow applications that are somewhat less relevant to our discussion, but are nevertheless interesting, appear in [JAD+94, GHS95, MC96, Sil95, DSW96, SJ96, SKM+96].

3.1 Scalability

Let us first discuss briefly three examples that point to different types of scalability needs. The first example is that of a somewhat traditional workflow application. It involves applying workflow and imaging technology in the Clark County Department of Business License for automating and streamlining the licensing system and to turn the department into a "paper-less environment"

[MC96]. ActionWorkflow [Act95] provided the workflow management infrastructure and Image X provided the imaging tools for this project.

Workflow requirements stemmed from the ineffectiveness of the previous system (which was primarily human-centered, paper based, and included an IBM ES 9000 mainframe program for information retrieval) to deal with the high demand for license renewals (approximately 72000 license renewals and applications annually). There was a need to automate the department and re-engineer business processes with emphasis on customer service. With a crude estimation of 250 working days, this gives around 290 instances of the workflow per day. While the 67 employees indicate the potential number of tasks involved in this workflow process, the number of tasks in the re-engineered process is unreported. The manual process for processing a request for general license that used to take 90-120 days took only 45 days after supporting a re-engineered process using the workflow and imaging technologies. While it seems that the current workflow products seem to support such medium scale applications, the question remains if the technology still imposes limitations that contributes to the 45 days period it still takes.

Service order processing, a telecommunications application, has been reported in [ANRS92, GHS95]. Unlike the previous example, operational system still does not use the modern workflow technology, and the workflow application was only written in prototypical form (compared to the commercial implementation in the previous application). It however gave a the following insights. The workflow involves interfacing with 15 to 25 legacy information systems (called Operation Support Systems). For special types of services such as a T1 line provisioning, a Bell company may only need to support tens of new workflow instances per day, while for more routine types of services such as POTS (Plain Old Telephone Service), thousands of new workflow instances would be generated. Because it takes several days in completing a service request, a WFMS may need to support tens of thousands of active workflow instances. In our views, three issues (and many other less important ones) that have hindered operationalization of workflow technology in this context are: administrative and managerial issues (mainly lack of management's understanding of the technology and its promise and the vested interests of those whose work could drastically change or be eliminated because of automation and reengineering afforded by the workflow technology), and concerns for robustness (including error handling and recovery) as any down time may not be acceptable and scalability (to support the high workloads).

Final application for our discussion in this area involves the use of workflows in a high-throughput mission-critical application system of tracking experimental data at the Center for Genome Research [BSR96]. Workflow automation is used in the laboratory information systems setting to automate the handling of samples, testing, instrumentation, data capture, and

tracking of event histories. The DBMS is primarily used in this project to control and track sample collection workflows.

The throughput of such experiments range in the order of approximately 15,000 transactions per day, with peak rates reaching 22.5 queries and updates per second [BSR96]. To be effective in such high-throughput production environments, a lot is desired in terms of scalability, efficiency, and reliability of the underlying WFMS infrastructure and the processing entities or resources that perform the high frequency tasks. The authors address the requirements of the DBMS that forms one of the critical components of the WFMS, and discuss LabFlow-1, a database benchmark for high-throughput production WFMS. Some of the important requirements for the DBMS mentioned in [BSR96] are:

1. standard database features such as providing isolation, consistency, failure recovery, high-level query language and query-optimization;
2. support for maintaining audit histories of the workflow activity (workflow tracking);
3. ability to store complex-structured data; and
4. ability to allow dynamic modification of the schema at run-time as the workflow itself is characterized by dynamism in terms of modification of flow of control and modification of tasks.

Two additional observations that are important with respect to the scalability in WFMSs are discussed next.

As a result of more automation and reengineering, number of instances of workflow that need to be managed would decrease. For example, if a workflow that used to take ten days will now take one day, than number of concurrent workflows to be managed for the same number of instances invoked will reduce by a factor of ten.

Perhaps the most important item related to the number of concurrent workflows is the tasks performed by humans. In general, a human performed (or manual) task would take significantly longer than a system performed (or automated) task and contribute to making the workflow instances long lived. Some of the human performed tasks could include the human involvement in error handling and recovery not handled automatically by the WFMS. Time it takes to complete some of the human performed tasks can also be unpredictable due to various reasons such as a decision making task involves consulting with someone else, the worker needs to attend to higher priority task or the worker goes on vacation. Some of the automated task can also take relatively long time, for example a task performed by a legacy system that uses a batch mode.

Now let us review some of the issues that are relevant to scalability from the perspective of a WFMS architecture.

The number of concurrent workflows, the number of instances of the workflows proceeded during a given period, and the average number of tasks in a workflow, all will have impact on the architectural issues.

Here is a non-exhaustive list of questions that would help us understand the scalability issues related to a WFMS:

- Do workflow modeling and design allow explicit allocation of tasks to different processing entities (that perform the tasks)? Do they allow specification of alternative processing entities? Does the design analysis tool help in analyzing these allocations?
- Is scheduling performed in a centralized manner or in a distributed manner?
- Are all relevant task managers (the processes on behalf of a WFMS that manage individual tasks) created as soon as the workflow instance starts, or only during some of the time (e.g., from the time its tasks has scheduling information to the time the task ends)?
- Do tasks of the same type share a task manager (i.e., do processes of WFMS components support multiple tasks, e.g., by using multi-threading)?
- What throughput limitations task managers impose? (As the task managers support more functionality, such as error handling and recovery, their size, complexity and I/O could increase by an order of magnitude or more).
- Does the WFMS support any form of load balancing, such as executing the tasks on alternative processing entities? Can it do so when alternatives are not explicitly specified in the design, either using run-time administrator or user intervention, or automatically?
- Can the processing entities or resources (e.g., DBMS in the third application above), which often predate the workflow applications and WFMSs, handle the load from workflow tasks (which may be new and in addition to other resource usage)? Do the operations from workflow tasks require synchronization?

In Section 4, we will discuss architectural and implementation decisions in an example WFMS that addresses many of these issues.

3.2 Adaptive and Dynamic Workflows

One of the most important of the demands evolving from new workflow applications is the ability of handling adaptable and dynamic workflows. Can a workflow (instance) and WFMS, through dynamic changes to its run-time environment, react and adapt to the rapid changes in process execution flow triggered by collaborative decision points, context-sensitive information updates, and other internal or external events? Some research issues in this area have been raised in the context of modeling and specification aspects appear in [HHSW96] and the relevant issues involving organizational changes appear in [EKR95, Her95]. However, the literature that addresses some of the run-time system issues is scarce.

Let us discuss some of the requirements using two classes of applications found in healthcare and defense, respectively. Example healthcare clinical applications are discussed in [SKM+96, HHSW96].

The first example concerns the issue of co-morbidity, which often arises in the healthcare environment. Here, a lab procedure, for example, brings out previously unknown disease or condition, which also needs concurrent treatment along with the original disease or condition. A physician, after considering the lab results, may decide to initiate an ad-hoc workflow in order to perform additional tests in view of the new findings. Note that the changes or new additions cannot be made independently of the original workflow because they relate to the same person, and management of one disease may affect that of the other. Since the amount of possible combinations of procedures is extremely large, it is almost infeasible to create a fully defined, all encompassing workflow for handling all possible cases. Therefore, a WFMS capable of supporting highly dynamic, adaptable workflows needed.

As another example scenario, consider the following defense application. Assume that a plan formulation workflow is currently developing a plan to move 5,000 troops from one point to another in a hostile environment. Such a process requires a variety of tasks performed by or directed by humans and a set of automated tasks, with appropriate control and data dependencies and constraints among them. Now assume that the intelligence monitoring activity determined a higher level of threat and the number of required troops is increased to 10,000. It would be necessary to consider many alternatives in a decision making process, as well as interruptions and redirections of the plan formulation process being enacted, such as (a) can the current process be used to support planning for the new requirement, (b) how far the current process has progressed, (c) if it is in the early phases, can it be interrupted and modified so that when completed, new requirement can be met, (d) should the current process that will result in planning for 5,000 troops be completed and can another process to support planning of another 5,000 troops be started, with support for appropriate interactions between the new process with the previous one, or (e) should the current process be prematurely stopped and a new process to address the additional requirement of 5,000 troops be started?

There may be wide range of events or conditions to which the workflows and a WFMS may need to adapt to, including the reasons of exception handling and load balancing. Being able to adapt through dynamic changes to workflows can also be an approach to develop a more robust and scalable WFMS.

A non-exhaustive list of questions that would help us understand the support for adaptable and dynamic workflows is as follows:

- Which types of events does the workflow needs to adapt to? Examples include
  - events resulting from execution of tasks
  - events from the changes in workflow specifications such as changes in persons or resources for a given role (e.g., due to change of work shift), or changes in the roles for given tasks (e.g., a new rule allowing a nurse practitioner to perform a task earlier requiring a physician to perform)

  - events from WFMS and its components
  - events from the infrastructure
  - events from external events including those from processing entities and resources or any aspect of the computing environment including user interventions
- What is the scope of the adaptation? Is the adaptation necessary for the workflow instance already in progress (i.e., is the workflow dynamic at the instance level)? Is the change applicable only to some of the active workflow instances or is it permanent for the workflow type? Will multiple versions of the same workflow type be managed and concurrently executed, and if so, which version should be used for the next invocation?
- How is the event conveyed to the WFMS (i.e., which protocols or APIs do monitoring agents or sentinels use to interact with the WFMS)?
- What are the ways in which a workflow may be dynamic (adding one or more tasks, adding additional conditions on or changing inter-task dependencies, adding new inter-task dependencies, creating a new version of a workflow type)?
- How do the changes interact or relate to on going instances of the workflows?
- How can you characterize a change to a workflow to be correct? Which criteria apply?
- Are the changes to a workflow introduced by a single task (an end-user or an automated task) or at the workflow administration level? Who should have the authority to introduce the changes and where?
- Since introducing changes dynamically may introduce logical errors to a workflow (or even a workflow type), what additional capabilities are needed in the area of error detection and recovery?

3.3 Integral Support for Collaboration

As noted earlier, work coordination, collaboration and information access are all key to providing comprehensive support for organizational activities. Researchers in group-ware and organizational systems seem to have taken early steps in integrating the first two by adding coordination or workflow capabilities to group support systems. However, many distributed and information systems issues still need to be addressed. Recent progress in video-conferencing and web-based collaboration (application sharing and white boarding) has rapidly expanded the availability of collaboration systems, and provided better opportunity for merging these formerly distinct capabilities.

To fully appreciate the importance of dynamic, collaborative workflows, consider the following examples.

For the first application scenario, let us further extend upon our health-care domain example. Suppose that in the process of a clinical workflow involving a primary physician, based on the available information, the physician decides to seek help of a consulting physician to determine the diagnosis.

The collaboration process may involve synchronous communication through video-conferencing, application sharing and data sharing (e.g., through Microsoft's NetMeeting or Netscape's Collabra), or may place asynchronous consultation request whereby the patient data and a videotaped message is provided to the consulting physician to seek an off-line help. These types of scenarios are being supported in our CaTCH (Collaborative TeleConsulting for Healthcare) prototype system (see http://lsdis.cs.uga.edu for further information).

Next, let us consider defense applications. Plan development for a large military operation is a very complex process typically involving scores of military planners using a variety of software systems. A computerized system created to support such a planning process must be highly dynamic, capable of supporting a wide range of collaboration among planners attempting to solve various problems at different stages of the plan development.

Suppose the workflow users in maritime operations are informed of a sudden crisis and decision to react to the problem has been made at the higher level of command. The maritime user, using her map display, understands the impact of this decision and schedules a new task to immediately contain the effect of the problem. The coordination system overseeing planning workflows notices the change in the task mix being executed and automatically forwards a change notice to the planning system managing the air-campaign. The automated planning workflow system immediately starts a new process to identify alternatives and informs the corresponding workflow manager to start a task to involve a human planner.

The air operations workflow management system interrupts the human planner by changing the color of the air campaign that will be affected by this change in his map display. The planner notices the color change, and interacts with the automatic planning system. The automatic planning system displays the various alternatives as different color coded information on the map. The human planner (in cooperation with maritime operations) then initiates a new process involving a set of new tasks, some involving tasks supporting automated planning and some involving manual coordination tasks with the maritime planner. If needed, the two planners may interact through a collaboration component (including video-conferencing) that would also assist the interactions by providing appropriate contextual information.

The above scenario calls for laughing a new process and modification to the existing processes. The new planning process may interrupt critical tasks in the maritime operations process plan or re-direct them in-order to deal with the crisis. It also involves creation of a set of recursively dependent processes that need to be coordinated at different levels of the coordination hierarchy. It would be desirable for the coordination system to manage the inter-dependent processes and assists the humans in understanding, interrupting, and redirecting them.

It is virtually impossible to fully account for all possible problems that might arise while coordinating the two independent planning workflows, as described above. Thus, the system must support other ways for seeking solutions. One possible way to achieve this is by "talking over", i.e. by conferencing between the appropriate commanders and planners.

Some of the question that need investigations are:

- How to uniformly model all objects (including resources) that represent coordination (tasks, sub-workflows, workflows) and collaboration (a video-conference session, white boarding, etc. with their time dimension)? How do you model dependencies for these two types of activities such as the following:
  - at some point in a workflow execution, a collaboration is started between humans and the workflow execution waits for the collaboration to end before it resumes, or
  - a collaboration activity designs a workflow and initiates it to watch how it is enacted
- In the above, is there a distinction between what constitutes coordination and what constitutes collaboration? What does it mean to lessen or remove that distinction in terms of a modeling paradigm?
- Can the result of a collaboration between humans be concretely and effectively captured to affect the coordination as appropriate (i.e., consistent with the decisions taken during the collaboration)?
- Is there a notion of correct execution that spans both these types of activities?
- When can you substitute a collaboration by a coordination (or vice-a-verse) and achieve perhaps better organizational effectiveness?
- How to model the collaboration that needs to be done at real time, as opposed to one that can be scheduled and deferred?

It seems that the level of technological integration is relatively easy, and some naive approaches may be adequate for certain applications. For example, we may simply treat collaboration as human tasks in a workflow design. However, some fundamental problems remain. What we believe is that on top of the *lower level* middleware supported by such technologies as Web, CORBA, database access APIs (ODBC, JDBC), etc. we will in the future have a *higher level* middleware, that provides the abstractions needed to seamlessly engage in coordination, collaboration, and information management activities. For brevity, we do not discuss the issues related to information management further.

4. METEOR$_2$ WFMS and its support for Scalability

This section presents the current state-of-the-art in WFMS with the METEOR$_2$ WFMS developed at the Large Scale Distributed Information Systems

Lab., University of Georgia (LSDIS) as the example. An obvious reason for using this system as an example is that we know it well. Another important reason is compared to discussing a commercial product or a laboratory prototype is that although METEOR$_2$ is a result of a research project, it also represents a commercializable technology that is being tested by or used in three organizations (CHREF, NIST and Boeing) outside of the LSDIS, with more expected in near future. Also, METEOR$_2$ utilizes the modern infrastructure technologies (specifically, Web and CORBA) and can be viewed as an example of how current state-of-the-art WFMs cope with the demands placed on the workflow systems today.

METEOR$_2$ has been geared towards developing a multi-paradigm transactional WFMS capable of supporting large scale, mission critical, enterprise-wide and inter-enterprise workflow applications in HAD environments. METEOR$_2$ includes all of the necessary components to design, build, deploy, run, and monitor workflow applications. Some of its key features include:

- Easy and quick application development: This is critical for the system's acceptance and organization's productivity. METEOR$_2$ graphical workflow designer enables an easy, point-and-click creation of the workflow process, associated workflow data, and task details that map to actual user and application tasks. More importantly, the workflow design process is followed by a mostly automatic code generation of an executable workflow application. In particular, all WFMS related run-time code (for scheduling and error-handling/recovery to the extent supported) is automatically generated. Naturally some of the tasks may need to be developed by human involvement or may already exist (i.e., legacy system supported task).
- Ease of use: End users can continue to use their existing user interfaces or be supported through a Web browser enabled GUI that the system can automatically generate.
- Support for heterogeneous tasks with transaction and legacy information system support: METEOR$_2$ model incorporates various task models have been defined to support integration of heterogeneous task types into the workflow model [KS95, MSKW96]. Task models include models for transactional, non-transactional, user and two-phase commit types of tasks. Ability to wrap legacy application and information systems is supported.
- Support for distributed and heterogeneous computing environment: Use of CORBA and Web-based environment support ability to develop distributed workflow applications running on heterogeneous computing environments.
- Error handling and recovery: These are integral to the METEOR$_2$ system with support for three-level object-oriented error handling and recovery model, and the ability to automatically support handling of many types of exceptions [Wor97].
- Scalability: METEOR$_2$ has been designed to handle workflow systems capable of carrying large loads. Specifically, we have considered a number of

workflow applications described in the literature as ones challenging the capabilities of the WFMS systems available today.

Several enactment services have been designed and implemented based on various scheduling paradigms [MSKW96]. These range from highly centralized ones to fully distributed implementations using CORBA and Web technologies (either exclusively, or in combination) as infrastructure for workflow enactment. Two distributed workflow enactment systems (ORBWork [DKM+97] and WEBWork [MPS+97]) have been successfully used to support a comprehensive prototype of workflow applications, including the statewide immunization tracking workflow process involving multiple hospitals and healthcare providers [SKM+96].

In the remainder of this section, we first present an overview of the METEOR$_2$ system. For the enactment service, we limit our attention to ORBWork. We also discuss scalability considerations in the context of ORBWork.

4.1 Workflow Design and Application Building in METEOR$_2$

The METEOR$_2$ graphical designer (MTDes) is used to develop a workflow application, in some cases leaving no extra work after a designed workflow is converted to a workflow application by the runtime code generator. MTDes is used to specify the entire map of the workflow, data objects manipulated by the workflow, as well as the details of task invocation. The workflow designer has the capability to model complex and varied tasks in a high-level conceptual and easy to use manner that shields the designer of the workflow from the underlying details of infrastructure or the runtime environment. It also aims at providing the user very few restrictions regarding the specification of the workflow. The designer assumes no particular implementation of the workflow runtime. Its independence from the runtime supports separating the specification aspects from the runtime.

MTDes has two design modes. The first mode, called *Process Modeler*, is aimed as a tool for the management of a typical enterprise. A workflow design may be initiated at the high, organizational level, without devoting any thought to the implementation details. This designer mode focuses on high-level specification issues without going into the procedural issues of the runtime. Thus the Process Modeler stresses the *what* of the process rather than the *how*.

The second mode, called *Workflow Builder*, assists in a GUI-supported development of a complete executable workflow application. The design specification created with the Process Modeler would be further refined in this mode. It is in this mode that we specify the entire map of the workflow including various tasks, task managers, and their interactions. This mode is intended for technical engineers or system analysts who would know the de-

tails of the underlying run time system and how the workflow application needs to be supported on it. The following discussion focuses on this mode.

The Workflow Builder has three components:

- the *map designer* for expressing the ordering of tasks and the dependencies among them,
- the *data designer* to model objects manipulated and transmitted by the tasks, and
- the *task designer* to provide details of the individual tasks (workflow activities) and interface to external task development tools (e.g., Microsoft's FrontPage to design the interface of a user task, or a rapid application development tool).

Details of MTDes and the WIL intermediate language are given in [Lin97, Zhe97].

4.2 The Runtime Code Generator

The workflow specification created using MTDes is stored in an intermediate format called the Workflow Intermediate Language (WIL). The WIL format is similar in structure and semantics to the Workflow Process Definition Language (WPDL) of the Workflow Management Coalition and supports most aspects of our earlier intermediate languages- Workflow Specification Language (WFSL) and Task Specification Language (TSL) [KS95]. A WIL specification includes all the predecessor-successor dependencies between the tasks as well as the data objects that are passed among the different tasks. It also includes definitions of the data objects, and the details of the task invocation details. Workflow definitions are stored in the workflow repository managed by the METEOR$_2$ repository service.

The functionality of MTDes has been tailored so that it allows for automatic code generation for a complete workflow application, except for the some of the individual tasks participating in a workflow. It is not possible to generate code for all of the computer (automated) tasks, since the individual tasks must be provided later by task developers (or possibly already exist as legacy applications).

Each runtime (ORBWork and WEBWork) has a suitable code generator. The runtime code generators work either using the WIL specification file as input, or using the repository service directly. The code generator outputs code for task managers, including their scheduling components, task invocation code, data object access routines, and the recovery mechanism. The code generator also outputs the code necessary to maintain and manipulate data objects, created by the data designer. The details provided using the task designer mechanism is used to create the corresponding wrapper code for incorporating legacy applications with relative ease.

4.3 METEOR$_2$ Runtime Support System

The METEOR$_2$ runtime architecture is centered around four major services: repository, enactment, monitoring, and error handling and recovery.

The METEOR$_2$ Repository Service is responsible for maintaining information about workflow definitions and associated workflow applications. The workflow designer, while using the graphical workflow design tool, communicates with the repository service and retrieves, updates, and stores workflow definitions. The designer is capable of browsing the contents of the repository and incorporating fragments (either sub-workflows or individual tasks) of already existing workflow definitions in the one being currently created. The repository service is also available to the enactment service (see below) and provides the necessary information about a workflow application to be started.

The METEOR$_2$ Enactment Service provides the necessary functionality for running workflow instances. The enactment service has two components responsible for activation of workflows and scheduling of tasks. The activation component allows the user to browse available workflow applications (using the repository service), select one of them, and then start a new workflow instance. The scheduling component of the enactment service is responsible for activating workflow tasks, once a workflow instance has been started. This part of the enactment service consists of the various task managers and their associated tasks, the user interfaces, the distributed error handling and recovery mechanisms, the scheduler (distributed among task managers) and the various monitoring components. As the web browser has become the preferred user interface tool, and has lead to Web-enabling of enterprise applications, we provide a browser-based interface for all end users to allow humans to participate seamlessly in workflows regardless of their location and (client) system platforms. Figure 4.1 shows the some of the key modules of the METEOR$_2$ system and their interactions.

The METEOR$_2$ Monitoring Service fulfills the function of an overseeing entity. It monitors all of the currently active (enacted and not yet terminated) workflow instances. The monitoring service interacts with the enactment service. It is responsible for detection of various types of failures that cannot be detected within individual sub-components of the enactment service.

The METEOR$_2$ Recovery Service interacts with the enactment and monitoring services. It is utilized once a failure has been detected and one or several workflow instances must be restarted. Additional details for error handling and recovery can be found in [Wor97]. The following sections describe the ORBWork enactment system and its associated recovery system.

4.4 The ORBWork Workflow Enactment System

As has been mentioned earlier, there is a host of different workflow management systems available in the commercial and the research arena. These sys-

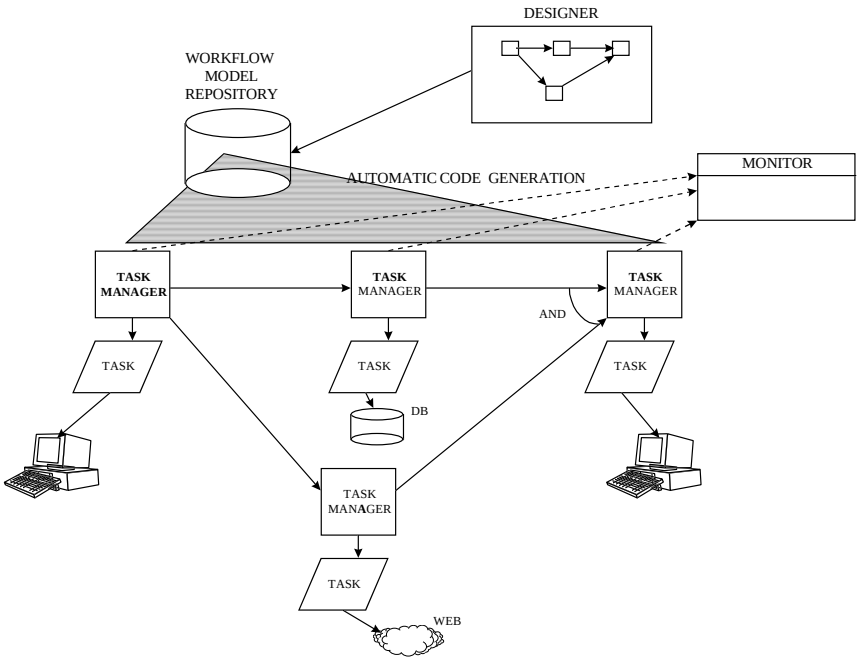

Fig. 4.1. The METEOR$_2$ Architecture

tems vary widely from their design philosophy and methodology and hence in their implementation strategies. The METEOR$_2$ model does not assume any particular architecture of the workflow enactment system. We have, however, opted to have a fully distributed architecture of the enactment system. Even though this introduces additional overhead and complexities a distributed implementation entails, a fully distributed architecture offers significant advantages for providing a highly scalable workflow management system.

Right from the start, our philosophy has been to build a system where each and every module has sufficient independence to perform its job efficiently but, at the same time, is also responsible to some evaluation entities in the organization (which in our case would be a workflow monitor). Aside from scalability, another significant driving force in favor of the distributed architecture is the absence of a single point of failure. This also has a profound impact on error handling and recovery issues.

ORBWork relies on CORBA to supply the basic communication infrastructure. This was almost the natural choice once we decided on a distributed framework, since CORBA provides an infrastructure for facilitating development of reusable, portable and object-based software in a distributed and heterogeneous environment. Currently, ORBWork is built with the use of Orbix and associated products (including OrbixWeb, OTS, etc.) from Iona

Technologies. (In our earlier prototype implementation, we had used Orbeline 2.0, now a Visigenic, Inc. product called VisiBroker for C++.)

All of the major components in ORBWork are implemented as CORBA objects. Also, it is common that tasks (or wrappers around existing legacy tasks) are also implemented as CORBA objects. As new CORBA services (in particular persistence and transaction) have become available in beta or product form (specifically from Iona), we have started to exploit them to reduce duplicating the corresponding functionality in the ORBWork implementation.

4.4.1 ORBWork's Distributed Scheduler and Task Manager Activation. The METEOR scheduler plays a central role in the ORBWork enactment service. Each workflow task has an associated task manager. The task manager is activated by the METEOR scheduler, once the task's input dependency is satisfied. The task scheduler is also responsible for activating the successor task schedulers, if any.

METEOR$_2$ does not have a single entity responsible for scheduling activation of task managers. Instead, the scheduling information is distributed among the individual task schedulers. The control dependencies, as defined in the workflow designer, are represented in the workflow process definition. The workflow code generator reads the workflow definition (retrieved from the repository) and automatically creates the task schedulers for each individual task in the workflow.

Each task scheduler has the necessary information about its immediate predecessor and successor tasks, and thus it is capable of activating the successor task schedulers once the task it controls terminates. (Ongoing work involves adding the capability to update this information at any time to support dynamic workflows.) Each task scheduler is activated by a group of immediate predecessor task schedulers. Such an activation provides all of the necessary parameters (workflow instance id, input data object ids, etc.) to schedule the task execution.

In ORBWork, each individual task scheduler is implemented (in C++) as a CORBA object. One such object handles the scheduling responsibilities for all of the workflow instances that invoke the task controlled by the scheduler and hence this component is perpetual.

The task activation part handles the execution of the associated task. The task scheduler simply activates the associated task manager, also implemented as a CORBA object, to run the actual task. After the invoked task manager terminates, it informs its task scheduler that the task has terminated. The scheduler then proceeds with scheduling the successor tasks, as defined in the workflow.

Communication between two task schedulers as well as between a task scheduler and the associated task manager is implemented by IDL method calls. In particular, when a task scheduler wants to activate a successor task (in fact the task scheduler of the successor task), it binds to the CORBA

object implementing the successor task scheduler and runs an appropriate IDL method to affect a transition.

METEOR$_2$ supports both automated tasks with no user involvement and human tasks involving participation of the end-users in a workflow application (e.g., to fill a form, validate data, carry out a task on a worklist, etc.). Each user task has a template of an HTML form associated with it. Such a template is first automatically created by the workflow designer. Each template contains information regarding which data object attributes a particular user task should display in the form.

While the automatically generated template uses a relatively simple layout, a template may be (and most likely will be, for commercially deployed workflow applications) refined by using any commercially available HTML authoring tool (e.g., Microsoft's FrontPage). When a workflow instance reaches a user task, the task's template is instantiated to include the field values taken from the data objects manipulated by this workflow instance. Specifically, the task manager of the user task retrieves the associated HTML form template, accesses the data objects mentioned in the template and then creates an instance of the template using values from the data objects.

Through this mechanism, METEOR$_2$ separates the specification of data objects necessary for user tasks from their actual use at runtime. However, the data must be transferred between the "CORBA world" and the end-user domain, an HTML form. METEOR$_2$ uses Java applets (ORBWork's implementation uses Iona's OrbixWeb) to retrieve the form's data fields and directly invoke IDL operations on data objects to write the new field values (data objects are also implemented as CORBA objects; see below). The applet's code is generated automatically by the METEOR's code generator and attached to the end-user's HTML form.

4.4.2 Handling Workflow Data. All of the data that is transferred among tasks in a workflow is represented as a collection of CORBA objects. The data object naming scheme associates each object to its workflow instance.

The data objects are specified with the use of the workflow designer and incorporated into the resulting intermediate representation (and stored in the worklfow repository). Then, the workflow code generator creates an appropriate IDL interface for each data object, as it generates code for the workflow definition.

Each data object supports the necessary functionality for providing its own persistence. This is implemented either with the use of an externally available persistent storage mechanism (for example an object-oriented database system) or by utilizing persistent object services offered by ORB.

4.5 Scalability of ORBWork

Scalability of the enactment system is one of the key requirements for workflow management systems today. We have leveraged the capabilities of the

ORB object's location independence that allows us to place task schedulers, task managers, data objects, and even actual tasks on separate hosts. The scalability has been addressed in ORBWork in several ways:

- During the design process, the designer may indicate specific server nodes where a particular task should be performed. The task itself need not be located on the same server as the controlling task manager. A task placement can be determined at application installation time.
- Typically, completing a workflow instance is a long duration process. Keeping this in mind, we adopted the *late activation* policy for task managers, tasks, and data objects. No data objects are created until the workflow instance needs them. Similarly, task managers are not created until a workflow instance reaches a specific task. Thus, the number of active objects (or ORBWork components) is kept to a minimum.
- Communication between task schedulers (and task managers) is limited to method invocations that are "light-weight". Only data object references are transferred between task managers (no large amounts of data are ever exchanged). This approach is in contrast to that used by the INCA [BMR96] model that passes process and workflow data between processing entities.
- Since ORBWork uses a fully distributed scheduler, there is no need for one, centrally managed scheduler. If necessary, each task scheduler may run on a separate host, allowing for the distribution of a potentially large load among many computers participating in a workflow. If the load carried by a single task scheduler is still too large, additional task schedulers may be added without much problem (a small amount of synchronization data must then be passed between task scheduler replicas). Each one would handle a portion of the overall number of workflow instances to be handled. A task scheduler supports multiple tasks instances corresponding to different workflow instances of the same type, reducing the number of CORBA objects needed for this purpose.
- The error handling and recovery framework for ORBWork (described in [Wor97]) has also been defined in a scalable manner by using error class hierarchies, partitioning the recovery mechanism across local hosts, encapsulating and handling errors and failures as close to the point of origination as possible, and by minimizing the dependence on low-level operating system-specific functionality of the local processing entities.

5. New Challenges to Support Dynamic and Collaborative Work

While researchers have discussed the needs and issues related to supporting dynamic workflows at the modeling and language levels [KS95, EKR95, HHSW96], there has been little discussion of the system support for such workflows. Workflow Management Coalition (WfMC) targets specification

and interoperability issues among current generation of products and do not yet support dynamic and flexible workflow processes.

In this section, we will replace the term workflow by *work coordination and collaboration process* (WCC process, or simply *work process* when there is less chance for ambiguity). A WCC process implies support for *dynamic and collaborative* organizational activities. Developing systems that could support WCC processes stands out as one of the difficult new challenges in future evolution of the WFMSs. Such systems must be uniquely sensitive to a rapidly changing process execution triggered by collaborative decision points, context-sensitive information updates, and other external events. Its distributed, reliable design must also ensure the robustness and scalability expected of large-scale systems that support WCC processes. We term such systems as Work Coordination and Collaboration Systems (WCCS).

5.1 Work Coordination and Collaboration System

We now present a conceptual architecture for a WCCS which we have started to design and develop as a follow-up to our work on METEOR$_2$. We hope that this architecture can be used as a source of many challenging directions for the research agenda in the collaborative workflow systems of tomorrow.

Unlike existing workflow management technology, WCCS must support multiple, event-driven execution paths as an integral part of collaborative problem solving. The system must be able to determine the process flow (possibly in real time) as decision points are reached, since a more active and integrated information substrate (consisting of integrated databases, modern distributed computing infrastructure and middleware, and monitoring agents, as discussed later) is simultaneously pushing relevant updates to the appropriate WCC processes. As in traditional workflow managers, the entire WCC process is modeled as activities and tasks for the people and system components participating in accomplishing the work. System components can be of various types, including traditional compiled and interpreted programs, wrapped legacy code or its invocation, database transactions, scripts, or emerging network computing.

Overall, WCCS provides an automated infrastructure for collaboration both within any high-level process and for all the interactions and dependencies among them. Support tools can be invoked in a number of ways: automatically, as an embedded part of a particular activity; identified to a user as a tool that is potentially beneficial for a particular task at some specified point during the execution of a WCC process; or user-selectable, and available on an ad-hoc basis.

To support the collaborative process at the information level, users must have a suite of information processing tools, or utilities, upon which to draw. These utilities can assist users with the execution of a range of activities, including planning, scheduling, analysis, filtering, browsing, and integration.

Like the collaborative technologies, these utilities can be associated with a particular task or can provide a general purpose service.

The new approach involves agent-based technologies to encapsulate functions, such as metadata management, ontological-based search, brokering, mediation, and sentinel services all of which are required to carry out the higher level cognitive processes in the environment. Agents are also tailored according to organizational role of users, for example, emulating the rules of a chain of command and providing appropriate support for authorization.

5.2 WCCS Design

The key WCCS capabilities include:

- specification and support for *dynamic processes* that can change at run time automatically or with human input. Here, dynamic refers to: a) the ability of the run-time model of the execution engine to change course automatically to support a newly-specified next step in the work process execution, b) work processes with automated application tasks as well as user tasks involving human involvement with structured or unstructured collaboration among participants, c) concurrent execution of (sub)processes corresponding to interactions among different tasks;
- a high degree *process reuse*, using a repository for consistency of process ontology, resource ontology (including user-centered ontologies such as organizational roles, authorizations, user profiling, and other context required to create the appropriate views of information in a particular task), work process definitions, and detailed task specifications;
- collaboration, providing a variety of tools (voice, video, whiteboarding) supporting human interaction, both at the task level and the work coordination level;
- *adaptable* work processes, using monitoring agents (also called sentinels), or other appropriate agents, that react to changes in the relevant information resources, fuse the appropriate information, and notify the work coordination activities. Here, adaptable refers to the ability of the WCCS to monitor, interpret, and react to rapidly changing information sources;
- *visualization* of concurrently executing work processes as well as data manipulated by the work processes;
- *security*, including authentication, traditional subject-object as well as task-based access control, and the active modeling and management of authorizations, providing Web browsers enabled with Java as the user interface for system administration (not discussed in further in this paper).

Although the WCCS runtime will enable automation of significant portions of the work processes using application (automatic) tasks, there will be continued need for involvement of humans in the process. Thus, processes supported by WCCS will involve heterogeneous tasks, including both human and automated (programmed or system) tasks. The work performed by humans

must be made easier and more productive. To this end, the tasks performed by humans in a work process can be enhanced with integrated collaboration support (including collaborative email, electronic discussion/whiteboarding, video/voice/data conferencing), and information/document search engine capable of accessing heterogeneous digital media (as in, for example, the InfoHarness system, see *http://lsdis.cs.uga.edu/infoharness*). The human collaboration will not be automatically managed or guided by the WCCS. However, WCCS will maintain an audit trail of the information processing and exchanges performed during a collaboration and enable such collaboration through appropriate Web-based tools.

The repository will contain process and information ontologies, and the tools to use them. It will consist of partial and complete designs and specifications of tasks, subprocesses and work processes. It will also support reuse of process components. An interesting feature it will support is the *plug-and-enactment* paradigm whereby process components in the repository can be used (through appropriate drag and drop support) on the visualization of a process being enacted to affect changes (such as interrupting, modifying, or redirecting) to the active processes. A unified WCCS user interface environment will support process design and development, browsing and querying of the repository, monitoring and tracking of processes being enacted by collaboration and enactment engine, and plug-and-enact interactions mentioned above.

Workflow process model supported by METEOR$_2$ is already quite comprehensive and sophisticated compared to contemporary WFMS products and WFMS research prototypes. It supports modeling of a hierarchical workflow process (with compound tasks), behavioral aspects of heterogeneous human-performed and application/system tasks (what can be observed, what can be controlled for a task execution by a given processing entity), intertask dependencies (with control and data flow), specification of interfaces (involved in supporting legacy applications, client/server processing and distributed processing), run time environment and task assignments to various system components, error handling and recovery requirements, etc. Primary enhancement are needed in the areas of modeling collaboration and specification of dependencies or interactions among tasks that support coordination and collaboration. Some of these can be adapted from the work group or CSCW systems research.

5.3 WCCS Architecture

The overall architecture of WCCS to support adaptive, dynamic, and collaborative work processes is composed of a number of participating WCCS components. A key WCCS component is a Work Coordination and Collaboration Engine (WCCE). In addition to WCCE, each component contains a Work Coordination and Collaboration Agent (WCCA), which is responsible for coordination of inter-WCCS activities. WCCS also includes one or more

Monitoring Agents (MAs), whose task is to monitor any observable changes in the external environment and signal them to the involved WCCAs. External changes are recorded in the (logically) Integrated Heterogeneous Media Database (IHMDB). A schematic of the WCCS architecture is shown in Figure 5.1.

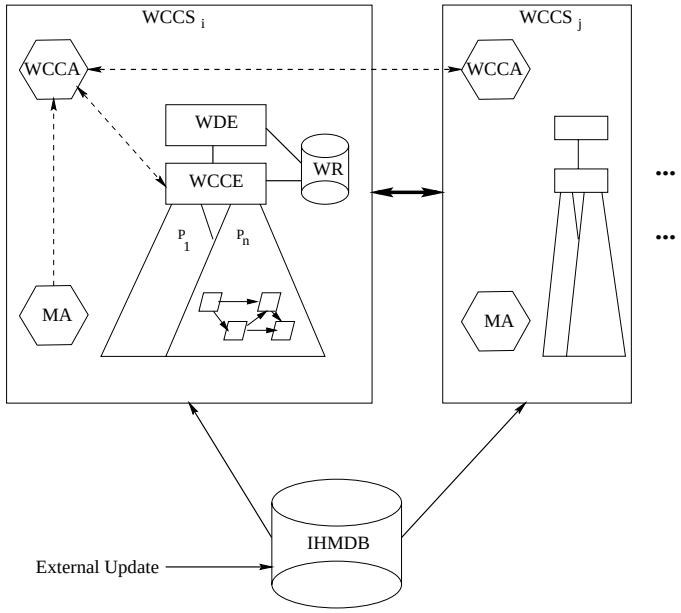

Fig. 5.1. Schematic of the WCCS Architecture

A WCCA oversees an associated WCCE. It can enact (possibly with a human involvement) a new work process instance and/or modify an already active one. A WCCA is also responsible for facilitating the inter-instance and inter-WCCS collaboration. In order to establish an inter-WCCS interaction, a WCCA receives a collaboration request from one of the tasks in a currently active instance. It then seeks out an appropriate peer WCCS and relays the request to the overseeing WCCA. Depending on the type of the request, the other WCCA may enlist an already active work process instance for participating in the requested collaboration, or enact a new instance.

The WCCS architecture also includes the Work Coordination and Collaboration Development Environment (WDE) and Work Coordination and Collaboration Repository (WR). To create the WDE component of the WCCS, we are planning to extend our current work to support the capability for multi-user design and a more dynamic and flexible model characterized above. Furthermore, WDE will have a tighter integration with WR and WCCE (as compared to the respective repository and run-time/enactment components

of METEOR$_2$) to support the *plug-and-enactment* paradigm. WDE, like other components of WCCS will also utilize the Java-based technology supported by the latest network computing and distributed object management infrastructure.

All the WCCS sub-components come with Web-based graphical user interfaces as well as protocols for open interactions developed on top of latest industry standards.

5.4 Work Coordination and Collaboration Engine

WCCE provides a general framework for the enactment and management of highly dynamic, collaborative work processes. As in the current METEOR$_2$, the main component participating in the scheduling and task activation are task schedulers and task managers. Collectively, the core functionality of a task scheduler and task manager is implemented as a Universal Task Manager Server (UTMS). A UTMS may dynamically accept a task definition and then participate in a WCCS supported activity to which the accepted task definition belongs. Task definitions may be defined or modified statically, for example with the use of our WDE tool, or dynamically, during the lifetime of a work process instance.

A UTMS communicates with other UTMS's and WCCA using Work Activity Transport Protocol (WATP). An external process, such as a WCCA (or possibly a Work Activity Monitor, a part of WCCE) may create a new UTMS at a desired host. A WATP control message may be sent to the new (or already existing UTMS). Such a message may contain a task definition (or re-definition), task migration, or possibly task termination information.

Once a UTMS has been established as participating within a given work process, it takes part in scheduling its associated task, as subsequent work process instances reach this particular UTMS. The scheduling is achieved by sending WATP scheduling (transition) messages from a UTMS to its successors, according to the currently existing work process map.

In addition to handling WATP protocol, a UTMS is capable of handling the HyperText Transfer Protocol (HTTP) requests. Thus the end-users in work processes can communicate with any UTMS by means of a typical Web browser. As such, a UTMS may be regarded as a specialized HTTP server with limited functionality.

A UTMS manages a number of HTML pages and forms related to the task it controls, as well as forms dedicated to general administration functions. Instead of involving numerous CGI-scripts, a UTMS simply runs a process to fulfill a given request. That process may be executed in UTMS's own address space, as an independent thread, or possibly as a heavy-weight process. In addition, we envision that a UTMS may communicate with the outside world with the use of other well known protocols, such as ODBC, JDBC, or possibly SMTP. In that sense, a UTMS may be regarded as a *protocol translator*. Figure 5.2 shows how a WCCE may be realized using UTMSs.

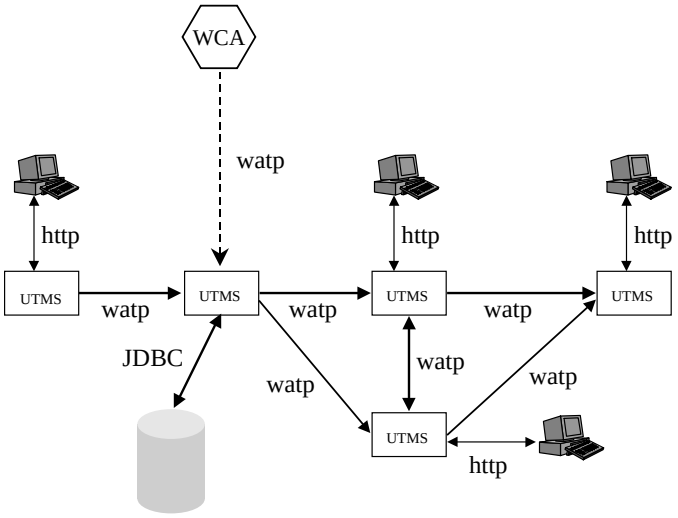

Fig. 5.2. Work Coordination and Collaboration Engine

A UTMS may be implemented in a variety of ways. However, we have chosen Java as it provides a portable, dynamic language. Because of Java's ability to dynamically load classes, implementation of dynamically loadable task definitions may be achieved quite easily. We plan to implement the WATP protocol on the CORBA/IIOP protocol.

6. Conclusions

Workflow technology has come a long way. Many routine (administrative) and ad-hoc processes found in an office environment can be supported by commercial products. Many organizations have achieved significant advantages using the workflow technology in terms of reduced paper work and faster flow of document-centric information (through electronic documentation and image flow systems), better quality (with more accurate information) of service to their customers (though faster access of electronic information, a side affect of some of the workflow automation), improvements through reengineering of processes, etc. As with the traditional 80-20 rule, 80% of simpler process can now be supported if there is enough will and resources on the part of an organization to use the workflow technology to achieve its advantages. Still perhaps 20% of processes– typically involving processes that can be described

as more complex, evolving, mission-critical, and large-scale– cannot be supported by current generation of products. These processes also tend to be higher value (as may be indicated by the adjectives "complex" and "mission critical").

Of the many outstanding technical challenges, we chose to address three of the more important technical challenges in this paper: scalability of a WFMS, adaptability through dynamic workflows, and support for collaboration. Arguably, these challenges range from shorter term to longer term in a research agenda we envisage. For these three challenges, we gave example application scenarios to provide motivations and to discuss some of the key requirements and technical issues. For scalability, we used the METEOR$_2$ WFMS to discussed how a modern system can deal with many aspects of this challenge. For other two challenges, we outlined the research approach for our work in progress or planned in future, and even speculated on ways to approach the issues when we reached the limits of our current experience and understanding. Our most important message is that coordination (as supported by current generation of workflow technology), collaboration (as supported by CSCW and work group systems) and information management will increasingly come together to merge into a higher level form of middleware.

Acknowledgements The METEOR team consists of Kemafor Anyanwu, Souvik Das, Yong Jiang, Krys Kochut (co-PI), Zonhwei Luo, John Miller (co-PI), Devanand Palaniswami, Kshitij Shah, Amit Sheth (PI), Devashish Worah, and Ke Zheng. Key past contributors include David Lin, Arun Murugan and Richard Wang. Special thanks for feedback from our industry partners who have tested and used METEOR$_2$ components at CHREF, SCRA, and NIST.

This research was partially done under a cooperative agreement between the National Institute of Standards and Technology Advanced Technology Program (under the HIIT contract, number 70NANB5H1011) and the Healthcare Open System and Trials, Inc. consortium. See URL: *http://www.scra.org/hiit.html*. Additional partial support and donations are provided by Iona, Informix, I-Kinetics, Boeing, Hewlett-Packard Labs, Persistence, and others.

References

[Act95] Action Technologies. Metro tour. Technical report, Action Technologies, Inc., 1995. URL: http://www.actiontech.com/.

[AAAM97] G. Alonso, D. Agrawal, A. El Abbadi, and C. Mohan. Functionalities and Limitations of Current Workflow Management Systems. Technical report, IBM Almaden Research Center, 1997. To appear in IEEE Expert.

[AKA+94] G. Alonso, M. Kamath, D. Agrawal, A. El Abbadi, R. Gunthor, and C. Mohan. Failure Handling in Large Scale Workflow Management Systems. Technical Report RJ9913, IBM Almaden Research Center, November 1994.

[AS96] G. Alonso and H.J. Schek. Research Issues in Large Workflow Management Systems. In *[She96]*, Athens, GA, May 1996.

[ANRS92] M. Ansari, L. Ness, M. Rusinkiewicz, and A. Sheth. "Using Flexible Transactions to Support Multi-system Telecommunication Applications," In *Proc. of the 18th VLDB Conference*, August 1992.

[BMR96] D. Barbara, S. Mehrotra, and M. Rusinkiewicz. INCAs: Managing Dynamic Workflows in Distributed Environments. *Journal of Database Management, Special Issue on Multidatabases*, 7(1):5–15, Winter 1996.

[BSR96] A. Bonner, A. Shruf, and S. Rozen. LabFlow-1: A Database Benchmark for High Throughput Workflow Management. In *Proc. of the 5th. Intl. Conference on Extending Database Technology*, pages 25–29, Avignon, France, March 1996.

[BDSS93] Y. Breitbart, A. Deacon, H. Schek, and A. Sheth. Merging Application-centric and Data-centric Approaches to Support Transaction-oriented Multi-system Workflows. *SIGMOD Record*, 22(3):23–30, September 1993.

[DKM+97] S. Das, K. Kochut, J. Miller, A. Sheth, and D. Worah. ORBWork: A Reliable Distributed CORBA-based Workflow Enactment System for METEOR$_2$. Technical Report UGA-CS-TR-97-001, LSDIS Lab, CS Department, Univ. of Georgia, February 1997.

[DSW96] W. Du, M. Shan, and C. Whitney. SONET Configuration Management with OpenPM. In *Proc. of the 12th Intl. Conf. on Data Engineering*, New Orleans, LA, February 1996.

[EL96] J. Eder and W. Liebhart. Workflow Recovery,. In *Proc. of the 1st. IFCIS Conference on Cooperative Information Systems*, Brussels, Belgium, June 1996.

[EKR95] C. Ellis, K. Keddara, And G. Rozenberg. Dynamic Changes within Workflow Systems. In *Proc. of the Conf. on Organizational Computing Systems (COOCS'95)*, 1995.

[Fis95] L. Fischer. *The Workflow Paradigm - The Impact of Information Technology on Business Process Reengineering*. Future Strategies, Inc., Alameda, CA, 2nd. edition, 1995.

[GHS95] D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119–154, April 1995.

[HHSW96] Y. Han, J. Himminghofer, T. Schaaf, and D. Wikarski. Management of Workflow Resources to Support Runtime Adaptability and System Evaluation. In *Proc. of PAKM'96, International Conference on Knowledge Engineering*, Basel, Switzerland, October 1996.

[Her95] T. Hermann. Workflow Management Systems: Ensuring Organizational Flexibility by Possibilities of Adaptation and Negotiation. In *Proc. of the Conf. on Organizational Computing Systems (COOCS'95)*, 1995.

[Hol94] D. Hollingsworth. The Workflow Reference Model. Technical Report TC00-1003, Issue 1.1, The Workflow Management Coalition, Brussels, Belgium, November 1994.

[JK97] S. Jajodia and L. Kerschberg, editors. *Advanced Transaction Models and Architectures*. Kluwer Academic Publishers, 1997.

[JNRS93] W. Jin, L. Ness, M. Rusinkiewicz, and A. Sheth. Concurrency Control and Recovery of Multidatabase Work Flows in Telecommunication Applications. In *Proc. of ACM SIGMOD Conference*, May 1993.

[JAD+94] S. Joosten, G. Aussems, M. Duitshof, R. Huffmeijer, and E. Mulder. *WA-12: An Empirical Study about the Practice of Workflow Management*. University of Twente, Enschede, The Netherlands, July 1994. Research Monograph.

[KR96] M. Kamath and K. Ramamritham. Bridging the gap between Transaction Management and Workflow Management. In *[She96]*, Athens, GA, May 1996.

[KS95] N. Krishnakumar and A. Sheth. Managing Heterogeneous Multi-system Tasks to Support Enterprise-wide Operations. *Distributed and Parallel Databases*, 3(2):155–186, April 1995.

[Ley95] F. Leymann. Supporting Business Transactions via Partial Backward Recovery in Workflow Management Systems. In *GI-Fachtagung Datebbanken in buro Technik und Wissenchaft*, Springer-Verlag, Dresden, Germany, 1995.

[LSV96] F. Leymann, H. J. Schek, and G. Vossen. Transactional Workflows, 1996. Dagstuhl Seminar 9629.

[Lin97] C. Lin. A Graphical Workflow Designer for the METEOR$_2$ Workflow Management System. Master's thesis, University of Georgia, Athens, GA, 1997. In preparation.

[MC96] R. Medina-Mora and K. Cartron. Action Workflow in Use: Clark County Department of Business License. In *Proc. of the 12th Intl. Conf. on Data Engineering*, New Orleans, LA, February 1996.

[MPS+97] WebWork: METEOR$_2$'s Web-based Workflow Management System. Technical Report UGA-CS-TR-97-002, LSDIS Lab, CS Department, Univ. of Georgia, April 1997.

[MSKW96] J. A. Miller, A. P. Sheth, K. J. Kochut, and X. Wang. CORBA-based Run-Time Architectures for Workflow Management Systems. *citedogac*, 7(1):16–27, Winter 1996.

[RSW97] F. Ranno, S. Shrivastava, and S. Wheater. A System for Specifying and coordinating the Execution of Reliable Distributed Applications. Technical Report, The University of Newcastle upon Tyne, England, 1997.

[Sil95] B. R. Silver. The BIS Guide to Workflow Software: A Visual Comparison of Today's Leading Products. Technical report, BIS Strategic Decisions, Norwell, MA, September 1995.

[She96] A. Sheth (ed.). Proc. of the NSF workshop on workflow and process automation in information systems. University of Georgia, May 1996. URL: http://LSDIS.cs.uga.edu/activities/NSF-workflow.

[SGJ+96] A. Sheth, D. Georgakopoulos, S. Joosten, M. Rusinkiewicz, W. Scacchi, J. Wileden, and A. Wolf. Report from the NSF Workshop on Workflow and Process Automation in Information Systems. Technical report, University of Georgia, UGA-CS-TR-96-003, July 1996 (also in ACM SIGMOD Record, September 1996). URL: http:// LSDIS.cs.uga.edu/activities/NSF-workflow.

[SJ96] A. Sheth and S. Joosten. Workshop on Workflow Management: Research, Technology, Products, Applications and Experiences, August 1996.

[SKM+96] A. Sheth, K. J. Kochut, J. Miller, D. Worah, S. Das, C. Lin, D. Palaniswami, J. Lynch, and I. Shevchenko. Supporting State-Wide Immunization Tracking using Multi-Paradigm Workflow Technology. In *Proc. of the 22nd. Intl. Conference on Very Large Data Bases*, Bombay, India, September 1996. A demo version of this application is at *http://lsdis.cs.uga.edu/demos*.

[SR93] A. Sheth and M. Rusinkiewicz. On Transactional Workflows. ACM SIGMOD Record, 22 (3), September 1993.

[TV95] J. Tang and J. Veijalainen. Transaction-oriented Workflow Concepts in Inter-organizational Environments. In *Proc. of the 4th Intl. Conf. Conf. on Information and Knowledge Management*, Baltimore, MD, 1995.

[Wor97] D. Worah. Error Handling and Recovery for the ORBWork Workflow Enactment Service in METEOR. M.S. Thesis, LSDIS Lab, Computer Science Department, University of Georgia, May 1997.

[WS96] D. Worah and A. Sheth. What do Advanced Transaction Models Have to Offer for Workflows? In *Proc. of Intl. Workshop on Advanced Transaction Models and Architectures*, Goa, India, 1996.

[WS97] D. Worah and A. Sheth. Transactions in Transactional Workflows. In *[JK97]*, chapter 1. Kluwer Academic Publishers, 1997.

[Wan95] X. Wang. Implementation and Performance Evaluation of CORBA-Based Centralized Workflow Schedulers. Master's thesis, University of Georgia, August 1995.

[Zhe97] K. Zheng. Designing Workflow Processes in METEOR$_2$ Workflow Management System. M.S. Thesis, LSDIS Lab, Computer Science Department, University of Georgia, June 1997.