A PROCESS DECOMPOSITION TECHNIQUE FOR DISTRIBUTED WORKFLOW MANAGEMENT

Giacomo Piccinelli

Hewlett-Packard Laboratories
Filton Road – Stoke Gifford
BS12 6QZ Bristol, UNITED KINGDOM

giacomo_piccinelli@hp.com

Abstract: Workflow Management Systems (WfMS) are expanding from Intranets to Extranets. The automation of inter-organization processes requires interaction between heterogeneous distributed systems, and coordination becomes an issue. While traditional solutions are based on strong centralization, we pursue a completely distributed approach.

In this paper, we propose a process decomposition technique for workflow processes. Starting from a generic workflow definition, we derive role-specific projections of the processes involved. Each projection is an autonomous process that embeds the (explicit) behaviour the original process assigned a specific role, and the (implicit) coordination logic for inter-role interaction. Projections can be distributed for execution on distinct process engines, and the network of projections reproduces the semantics of the original workflow process. The enactment environment for the projections is a lightweight infrastructure we implemented using Java RMI.

Keywords: Distributed workflow, process coordination, role management

1 INTRODUCTION

Workflow Management Systems (WfMSs) represent a category of software tools specifically designed to support process automation within complex organizations [14]. Procurement, production control and expense claims are significant examples for the type of processes to which WfMSs apply. The resources involved range from automatic systems (like sensors, computers and telecommunication systems) to human beings. The distribution level for the resources ranges from small and

L. Kutvonen et al. (eds.), *Distributed Applications and Interoperable Systems II*

medium distances (different offices in the same building, different parts of a factory) to very big distances (offices in different countries) [7,9].

Communication is the base for cooperation, and the evolution of network technology is having a big impact on WfMSs [1,5]. Improved connectivity not only allows better execution for existing solutions. Processes are (re)engineered in order to take advantage of the opportunities offered by new technologies. Just-In-Time (JIT) procurement is a significant example. Process design blends procedural knowledge (on *how* to obtain a result) with operational knowledge on the infrastructure that will support the execution. By providing the designers with an abstraction of the execution infrastructure, WfMSs allow them to focus on the procedural aspects of the process [2,3].

In our work, we focus on inter-organization workflow management. Starting from a standard process definition language [14], we enrich it with some basic features oriented to cooperative environments [4]. We then propose a process decomposition technique that automatically generates role-specific projections for the cooperative process. Each projection embeds the (explicit) behaviour that the workflow process assigns to a specific role, and the (implicit) coordination logic for inter-role interaction. Each organization requires only the projections for the roles it plays in the cooperative process.

Organizations usually have an internal WfMS (sometimes more than one) [22]. Therefore, the options for the enactment of the projections lay basically along two major lines: translation and bridging. Translating a projection into an internal process for each existing WfMS can improve efficiency, but it requires specific conversion tools for every possible WfMS. The alternative is to introduce a new basic WfMS in each organization, and executing the projections on it. The existing WfMSs inter-operate with the new one via a predefined interface (bridge). Balancing performance, generality and implementation complexity, we adopted the second alternative. The bridging interface is modeled on a workspace metaphor. The projection enactment environment is a lightweight infrastructure entirely implemented using Java.

This work is part of a project called RABBIT (Research on Advanced Business-to-Business Information Technology) on which the author worked for the past two years. We first present the cooperation metaphor (workspace) and the process definition language (PDL) used in RABBIT. The description of the projection algorithm and its integration in the overall system represent the central part of the paper. We then present a brief overview on related works, and our conclusions.

2 THE WORKSPACE METAPHOR

Workflow processes tend to be quite simple, in terms of both process and data flow: the focus is on precision. Data units are files or very simple data structures based on strings and integers. The process flow is shaped on the evaluation of simple conditions, like the termination of a task or the test of a value against a given threshold. Workflow processes have strong implications on the economy of an organization, and their technical simplicity reflects the needs of reliability and clarity. Process designers usually have administrative, managerial or technical (non-IT) expertise. They have to concentrate on the actual needs of the organization (also referred to as *business logic*), not on technology. If a metaphor is used, it has to be

close to the way a process designer perceives the organization. There is no space for ambiguity, and all the assumption made in the model have to be clear and acceptable to the designer.

The standard metaphor for WfMSs is the work-list, which is a legacy from the first (paper based) implementations of workflow management concepts. A step in the workflow results in the posting of a folder (containing data and instructions) in the "in-tray" of the resource(s) in charge of the processing of the data. A work-list can be imagined as a stack of folders, and the high-level behaviour of a resource is to process the content of the folders in the associated stack(s). The result goes in the "out-tray", and it is automatically sent to the resource(s) in charge of the next step. The definition of a process consists in the specification of the content of the folders, and the path they follow. Blackboard metaphor is on the same line. The difference is that blackboards now replace the trays, and all the folders are pinned to the blackboard. On the cover of the folder there are the indications on who should process the content [11].

Peculiar aspects in inter-organization processes are security and autonomy. Cooperation implies that organizations share part of their resources and expertise, but it has to be done in a controlled way. The *workspace metaphor* we propose specifically addresses these issues. Each organization is associated with a data repository called *workspace component*, which represents the view the organization has of the cooperative activity.

Objects are the result of an activity ("artifact" or "work item" [14]). In the *object space*, the organization places the artifacts it shares with its partners, and retrieves the artifacts they produced which are relevant for the execution of its internal tasks. The organization has immediate access only to the artifacts in its object space, but these are also the only resources potentially exposed to its partners. Each organization shares all and only the objects it agrees to release and under the circumstances defined in the cooperation process. At the same time each organization receives all and only the objects it is entitled (requested) to work on. Messages represent information on the state of either the system or the process. Tasks are atomic operations like the execution of an activity [14] or the manipulation (insert, withdraw, process) of artifacts and messages. *Task space* and *message space* are managed in a way similar to the object space. The global *workspace* is the union of all the workspace components.

The links between our model and the work-list model derive mainly from the fact that each organization has a private logical space for information exchange. From the blackboard model we take the planar distribution of the items in that space. Especially for the tasks, the unstructured planar distribution emphasizes the parallelism and non-determinism in the execution sequence. During the execution of a process, the WfMS enforces the fact that all the tasks in the task space of an organization can be executed in any order and possibly in parallel.

For messages and objects, it emphasizes their direct availability to whoever needs them in the organization.

3 PROCESS DEFINITION LANGUAGE

The basic operations in a cooperative process are related to the exchange of artifacts, messages and services. The scope of the activities can be either limited to a single organization (task) or to more than one organization (service).

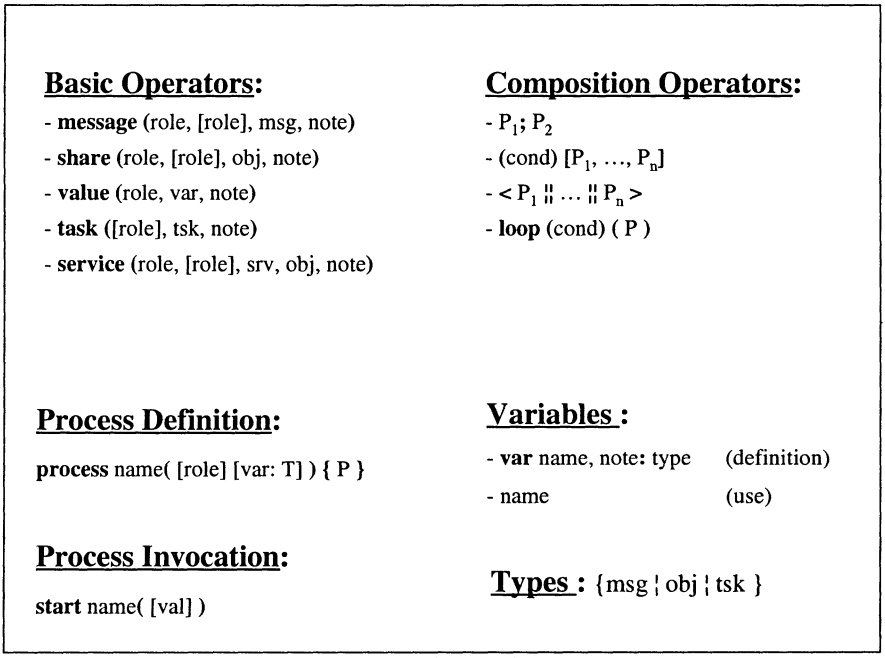

Figure 1. Process Definition Language.

Purpose of this section is to give a general understanding of the features provided by the process definition language (PDL) to the process designer. Languages like Hoare's CSP [13] and Milner's CCS [17] had a strong influence on our formalism, but we explicitly target the peculiarities of cooperative workflow. Starting from a standard PDL [14], the language was enriched with features specific to the modeling of cooperative environments. The resulting language (Figure 1) offers complete abstraction from the distribution problems. The designers can concentrate on the definition of the process as a whole, focusing only on the behaviour of the organizations. The actual semantics of the entire language has been formalized following an approach (operational style) similar to the C-FAM (concurrent functional abstract machine) [10].

The process designer acts like an independent coordinator. The organizations are modeled as macro-resources with very simple capabilities. An organization can share (*share*) data with its partners, as well as send them control information (*message*). A *task* defines aspects of the process of direct interest for an organization. Service exchange (*service*) enforces the central role of cooperation among the organizations.

An organization can also be required to provide the content (*value*) of process parameters.

The management of the process flow is based on four basic options: sequence, loop, parallel and alternative paths. The *sequence* operator ";" indicates that all the tasks in the process P_1 need to be completed before starting any task indicated in P_2. The overall process (P_1; P_2) ends when P_2 ends. The *parallel composition* operator allows multiple execution threads (sub-processes) within a process. The global process $< P_1 \,||\, \ldots \,||\, P_n >$ is completed when all the sub-processes P_i are completed. In the *choice* operator, one and only one process among the P_i is executed depending on the evaluation of the expression *cond*. The evaluation of *cond* returns an integer value *k* in the range 1...n, and only P_k is executed. The overall process is complete when P_k is completed. The loop operator allows cycles in the process flow. The basic idea is to re-execute P until the guard condition becomes false.

Each module (*process*) represents a process in which some of the values are left unbound. In particular, we use the concept of *role* [20]. A role may be considered as an identifier for one ore more organizations. It will be bound to an actual set of organizations at execution-time. Other parameters can be messages (msg), objects (obj) or tasks (tsk). Simple and mutual recursion is allowed, while nested definitions are forbidden at the moment.

4 PROCESS DECOMPOSITION

A cooperative workflow process involves different entities we generically refer to as organizations. Autonomy and independence are basic characteristics of a real-world organization, and they should reflect on the way processes are designed and managed. Different organizations require different views on a process, different knowledge on the parties involved, and possibly different quality of service (QoS) from the enactment infrastructure. The identification of the roles involved in a process and the isolation of their expected behaviour is a major step towards the satisfaction of this kind of requirements.

A role indicates an entity (or a group of entities), which can be associated with a specific set of actions (behaviour) in the process. In the process for the approval of a new research project, for example, we can think of funding organizations, proposal reviewers and the proposal initiators as some the roles involved. The process designer perceives the set of roles as a resource repository to organize into a global process. The single organizations perceive the process in terms of the roles they play in it, and the work deriving from these roles. The idea behind our work is to conciliate the needs of the designers with the expectations of the executors of the process. Global design, customized execution.

The idea of defining a process and then assigning roles to organizations is common practice. The options for the enactment of the process range from complete centralization (one process on a single WfMS) to complete distribution (each organization creates an internal process, which runs on its own WfMS). The more the organizations are interested in autonomy and independence, the more the solutions shift towards the distributed option. The practical problem is that projecting a process on the various roles (organizations) involved is a complex task. A totally new layer of synchronization and flow management solutions needs to be introduced in order to preserve the semantics of the original process.

The purpose of our work is to provide automatic support for role management and process partitioning (projection).

We first present the logical model of the enactment infrastructure for the projections. We then propose and discuss the process decomposition algorithm. A description of how the projection generator and enactment environment have been implemented in the RABBIT system closes this section.

4.1 Enactment Model

The idea we propose for the projections is the one of fully-fledged processes, in which process logic blends with coordination logic. The enactment environment is based on the workspace metaphor, and the existence of a basic communication layer is assumed. We also assume the existence of workflow engines (part of a WfMS in charge of the execution of a process) able to understand the process language P-PDL used for the projections (Figure 2). In order to keep the presentation of the model simple, we assume that each organization is associated with a distinct workspace component and a distinct process engine. Different configurations can be derived from this basic model, and the impact on the actual infrastructure is minimal.

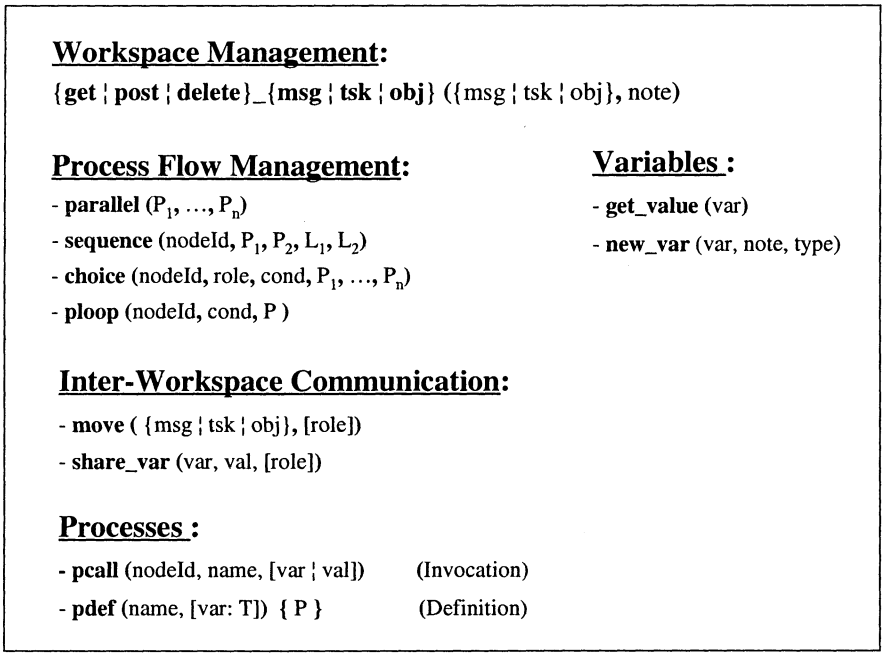

Figure 2. Projection-level process definition language (P-PDL).

The projections related to all the roles an organization plays are executed on the same engine. The three major areas of activity for the engine (Figure 2) are: workspace management, process flow management and inter-organization

communication. The workspace is the interface between the engine and the organization. On the input side, the engine posts in the workspace the tasks to be performed, as well as objects and messages coming from other organizations. On the output side, the engine collects objects, messages and task description to be delivered to other organizations. The management of variables deals with the access to the internal state of a process instance.

Inter-workspace communication actually maps to engine-to-engine (therefore organization-to-organization) interaction. Objects, tasks and messages that the engine collects from the workspace can be sent to other engines. The value of a variable can be sent to other engines in order to preserve the consistency in the state of the projections. If more than one organization is involved in a sub-process, they all need to keep a consistent view on its state.

Managing the flow of the process is the core activity of the engine. Consistency between different organizations has to be maintained at each stage, and all the operations are performed in a transactional style. The field *nodeId* in the parameter list of the operators represents a unique identifier indicating the static position of the operation in the abstract-syntax tree [8] of the process. The engine combines the static information (nodeId) with execution-time information, and the result allows the unique identification of points in the execution tree [8] of the processes. All the engines can compute the unique identifiers independently, and they can use them as a common reference (e.g. for synchronization).

The semantics of *parallel* in the P-PDL (Figure 2) is the same as in the PDL (Figure 1). The branches can be executed at the same time or in an interleaved way. The semantics of *choice* requires the organization to proceed with the execution of only one of the branches of the process, depending on the result of the evaluation of the condition (*cond*). The choice of the branch has to be consistent among all the organizations involved in any of the possible branches. This is an instance of the consensus problem [18], which we address imposing the election of an organization in charge of the decision. As the structure of the conditions to evaluate in a workflow process usually depends on the peculiarities of the problem, we do not impose any predefined structure on cond. A specific organization (called evaluator) receives the evaluation of the condition as a task to perform, and the result is then distributed to the other organizations involved. The evaluator can be indicated explicitly in the condition itself. Otherwise there is a random selection among the organizations involved in the choice. The semantics for the loop (*ploop*) is standard; the assumptions made for the condition (cond) in the choice operator still apply. The semantics of the *sequence* operator addresses the synchronization problems introduced by the existence of multiple process threads (projections). For the semantics of ";" in the PDL to be preserved, an organization needs to know the identity of the organizations involved in P_1 (list L_1) and in P_2 (list L_2). The organization will wait until all the members of L_1 complete P_1 before starting P_2. In the same way, the organization notifies all the members of L_2 when it completes P_1. Process definition (*pdef*) and invocation (*pcall*) have standard semantics.

4.2 Process Decomposition Algorithm

The process decomposition procedure has two basic components: role searching and projection generation. The assumption for the roles is that they are organized in a

dependence hierarchy H. Based on H, the binary relation *"derives"* is provided. The properties required for this relation are reflectivity (-x derives x-), transitivity (-x derives y- and -y derives z- implies -x derides z-) and anti-symmetrical (-x derives y- implies that -y derives x- is false). The relation is defined together with the process. The role searching activity consists in the scanning of the process definitions in order to match the roles defined in H with the one used in the code for the sub-processes. Additional information is collected during that phase, which is used for the optimization of the projection procedure.

The projection generation isolates the parts of the process that involves a specific role, and maps them into their P-PDL (Figure 2) equivalent. The projection algorithm is function $P_x(\ )$: PDL → P-PDL. The recursive structure of the PDL allows a modular definition of $P_x(\ )$. The formal specification of the entire algorithm would take more than the space allowed for the entire paper. We try anyway to capture its structures, and the concepts it is built upon.

The basic idea is to consider the logical position of a role (its syntactic name) in the PDL source code. Given the characteristics of the PDL operators, it is always possible to identify the activity (in terms of workspace management and mutual synchronization) that an organization playing that role has to do in order to preserve the semantics of the cooperative process. A very simplified mapping for sharing operator, for example, could be:

$$P_x\ (\textbf{share}\ (role_S, [role], obj, note) = \begin{cases} \textbf{get\_obj}\ (obj, note) & \textit{if } X = S \\ \textbf{post\_obj}\ (obj, note) & \textit{if } X \textit{ in } [role] \\ nil & \textit{otherwise} \end{cases}$$

The engine which executes the projection for the organization playing the role S, (sender) it places a request for the object in the workspace of the organization. The engines supporting the organizations associated to the roles in the list [role], they post the object in the object space of the organizations in order for them to collect it. The organizations associated to other roles do not have any notion of this exchange.

The actual mapping for share operator is slightly more complex, as we need also to instruct the engines to move the object from one to the others as well as deleting the object from the workspace when no longer required. The mapping for choice operator of the PDL can be as follows:

$$P_x\ ((cond)\ [P_1, \ldots, P_n]) = \textbf{choice}\ (nodeId, role, cond, [P_x(P_1), \ldots, P_x(P_n)])$$

NodeId is the position of the operator in the static-syntax tree of the process. The role is the one of the evaluator, specified in the condition (cond) or selected randomly among the roles involved in the branches of the choice. We enforce a pull approach for the synchronization of the roles. The engine of the evaluator maintains the result of the evaluation of cond, and when the other engines need that result they ask for it. The condition may be in the body of loop, or there may be recursive function calls. Using the nodeIds (present in all key operators of the P-PDL) and other run-time information, the engine builds a unique run-time identifier for the execution point to which the result of the evaluation refers to. This technique solves the ambiguity problems also when the same condition is evaluated on different

parallel branches of a process. The projection function is applied recursively to each branch of the choice operator.

Specific functions have been defined in order for example to establish if a role actually uses a variable, therefore if needs to receive the updates on its value. The focus is in general on minimizing the synchronization activity, while preserving the semantics of the process. This problem is evident in the sequence operator. Optimizations on the composition of the list L_1 and L_2 (Figure 2) of the organizations to be synchronized before moving from P_1 to P_2, can have a big impact in terms of performance. The time scale in a workflow process is considerably bigger than normal distributed applications, but there are a lot of process instances going on at the same time. Static analysis on the program is a starting point, but execution-time solutions can make a big difference.

4.3 Enactment Infrastructure in RABBIT

Most of the ideas presented in this paper about the workspace metaphor and process decomposition are at the base of the distributed infrastructure of the RABBIT workflow management system. The PDL compiler and the distributed enactment infrastructure are the main components of the WfMS. The system has been entirely developed using the Java Platform™, and in particular the Remote Method Invocation (RMI) package.

The compiler works in two phases: role detection and projection generation. The compilation unit is the *project*, which is composed by a set of process definition files. During the first phase, the compiler performs a scan of the project in order to identify the roles involved in the process. In the second phase the projections of the entire project are generated and stored into files (one per role). The file contains the P-PDL source code for the workflow engines, plus control information that the engines use for initialization of the run-time process instance.

The main components of the actual enactment infrastructure are the engines and the workspaces. Engine and workspace have been developed as two different applications, therefore they can be installed on different machines. The workspace can actually be installed as a single application or as three separate modules. The elements of the object space can be quite big (e.g. documents or pictures), and it might be useful to keep them separate from messages and task descriptions. In terms of security, the system supports replication for both the engines and the workspaces. The transactional features of the content update procedure guarantee the consistency between the copies.

A single engine can host instances of the projections of more than one organization, and it can have connections with more than one workspace. If a single organization has more than one workspace, the data flow generated by different processes can be routed to different workspaces.

5 ALTERNATIVE AND COMPLEMENTARY APPROACHES

Process specification formalisms range from Petri Nets (SPADE) [2] and similar graphtal approaches, to imperative object oriented models (EPOS or SOCCA) [9]. The main categorization contrasts event based solutions like Process Wall [11] and, procedural based solutions like Exotica [1]. The cooperation paradigms range from

direct message exchange [15, 19] and its variations [1] to blackboard based solutions [11,21].

Process-centered environments [23] and workflow systems [14] are under pressure due to the explosion of Internet technology. Web enabled systems [16] are a first answer to the problem but more specific solutions [4,6] need to target network based environments. Different options are open for distributed inter-organization processes [7], and the right mix depends on environmental constraints. Interworkflow [13] for example, offers visual support for the manual definition of process projections and WfMS-specific translation.

The importance of cooperation between heterogeneous distributed WfMSs has been recognized by international organizations like the WfMC (Workflow Management Coalition) and the IETF (Internet Engineering Task Force). Together, they are working on an open protocol (SWAP – Simple Workflow Access Protocol) for WfMS interoperability. While the Internet provides physical connectivity, SWAP will provide process-level connectivity.

6 EXAMPLE

In this section, we discuss the application of our approach to a project approval scenario involving different organizations. We refer to a very simple Project Approval process (Figure 3), and we focus on the generation and enactment of its projections. The process involves three roles: proposer, reviewer and sponsor (we do not discuss the Grant sub-process).

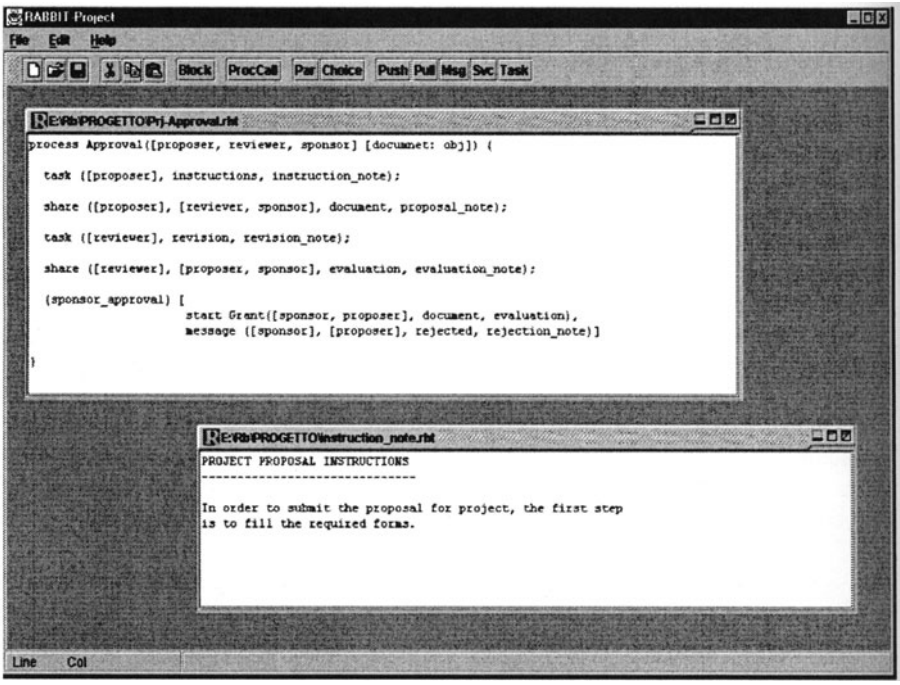

Figure 3. "Project Approval" Example.

The RABBIT compiler automatically detects the roles involved, and generates three projections. The sponsor, for example, receives the project proposal, then the project review, and only at this stage it starts the decision process. This is the enforcement of a "good practice" for the sponsor, but is also a guarantee for the proposer. The synchronization enforcing the semantics of the process is automatically and consistently deployed into all the projections. Projections can be enacted on different engines; therefore if one organization experiences temporary problems (and there are no process-related dependencies) the other organizations can carry on with their work. As an example, the computer hosting the projection of the proposer can be isolated from the network while the reviewer is sending its results. This doesn't prevent the sponsor from evaluating the sponsor_approval condition. Anyway, if the project is rejected the sponsor waits until the proposer can receive the appropriate notification before closing the process.

Projections are used also for enhanced security management. If more than one projection is hosted on the same engine, the granularity of the security mechanisms (like ACL – Access Control Lists) can be considerably improved. As projections are based on roles instead of organizations, different organizations can play different roles in different instances of the process. The possibility to change the roles assigned to the organizations at run-time (dynamic roles) is under investigation.

7 CONCLUSIONS

In this paper we presented a decomposition technique for the automatic identification of the roles involved in a cooperative workflow process as well as the behaviour expected for these organizations responsible for these roles. Starting from a single cooperative process, we generate role-specific processes (projections). A projection embeds all the process logic for the specific role, plus inter-role synchronization logic. Projections are autonomous processes; therefore they can be distributed transparently. The interface between a projection and the external world is shaped on a workspace metaphor (also described). The ideas presented in the paper implemented in RABBIT, a Java-based workflow management system we developed.

In addition to process distribution, we see role separation as an enabler for quality of service (QoS) and security management. On the security side, each organization has a specific view of the cooperative process depending on the role(s) it plays. If an organization plays multiple roles, it can manage each of them in different ways. On the QoS side, the isolations of the roles can be used to provide differential services as well as fine-grain resource allocation. An extensive exploration of these possibilities represents the main development line for our work.

References

[1] ALONSO G., MOHAN C. AND GÜNTHÖR R., Exotica/FMQM: a persistent message-based architecture for distributed workflow management. *Proc. IFIP Working Conf. On Systems for decentralized organizations*, 1995.

[2] BANDINELLI S., DI NITTO E. AND FUGGETTA A., Supporting cooperation in the SPADE-1 environment. *IEEE Transactions on Software Engineering*, Vol. 22, no. 12, 1996.

[3] BARGHOUTI N., Supporting cooperation in the Marvel process-centered SDE. *Fifth ACM SIGSOFT Symposium on Software Development Environments.* Herbert Weber (ed.), 1992.

[4] BASILE C., CALANNA S., DI NITTO E., FUGGETTA A. AND GEMO M., Mechanisms and policies for federated PSEEs: basic concepts and open issues, *Proc. 5th European Workshop on Software Process Technology.* Nancy, 1996.

[5] BEN-SHAUL I., COHEN A., HOLDER O. AND LAVVA B., HADAS: A Network-centric framework for interoperability programming, *Proc 2nd Inter. Conference on Cooperative Information Systems*, 1997.

[6] BEN-SHAUL I AND KAISER G., Federating process-centered environments: the Oz experience, *Automated Software Engineering, Vol. 5.* Kluwer Academic Publisher, 1998.

[7] BEN-SHAUL I. AND KAISER G., Integrating groupware activities into workflow management, *Proc. 7 th Israeli Conference on Computer Based Systems and Software Engineering*, 1996.

[8] COHEN B., HARWOOD W. AND JACKSON M., *The Specification of Complex Systems,* Addison-Wesley, 1986.

[9] FINKELSTEIN A., KRAMER J. AND NUSEIBEH B. (Ed.), *Software process modelling and technology*, Research Studies Press, 1994.

[10] GIACALONE A., MISHRA P. AND PRASAD S., FACILE: A symmetric integration of concurrent and functional programming, *Proc. Of TAPSOFT'89, Vol.2. Lecture Notes in Computer Science (LNCS 352).* Springer-Verlag, 1989.

[11] HEIMBIGNER D., The Process Wall: a process state server approach to process programming, *Proc. 5th SIGSOFT Symposium on Software Development Environments*, 1992.

[12] HIRAMATSU K., OKADA K. AND HAYAMI H., Interworkflow System: Coordination of Each Workflow Systems Among Multiple Organizations, *Proc. 3rd IFCIS Int. Conference on Cooperative Information Systems*, New York, 1998.

[13] HOARE C., *Communicating Sequential Processes*, Series in Computer Science. Prentice-Hall, 1985

[14] HOLLIGSWORTH D., The workflow reference model. *Workflow Management Coalition (WfMC)*, TC00-1003, 1994.

[15] KRISHNAKUMAR N. AND SHETH A., Managing heterogeneous multi-system task to support enterprise-wide operations, *Distributed and Parallel Databases.* Kluwer Academic Publishers, 1995.

[16] MILLER J., SHETH A., KOCHOUT K. AND PALANISWAMI D., The future of Web-based workflow, *Proc. of the International Workshop on Research Directions in Process Technology.* Nancy, 1997.

[17] MILNER R., *A calculus of communicating systems. Lecture Notes in computer Science* Vol. 32. Springer-Verlag, 1980.

[18] MULLENDER S. (Ed.), *Distributed Systems.* ACM Press, New York, 1993.

[19] OBJECT MANAGEMENT GROUP (OMG), *CORBA facilities: common facilities architecture V4.0*, OMG 1995.

[20] SANDHU R. AND PARK J., Decentralized User-Role Assignment for Web-based Intranets, *Proc. 3rd ACM Workshop on Role-Based Access Control*, 1998.

[21] SCHWARTZ D., *Cooperating Heterogeneous Systems.* Kluwer Academic Publisher, 1995.

[22] SHETH A. (Ed.), Report from the NSF workshop on workflow and process automation in information systems, *Proc. NSF workshop on workflow and process automation in information systems: state-of-the-art and future directions*, 1996.

[23] WOLF A. AND ROSENBLUM D., Process-centered environments (only) support environment-centered processes, *Proc. 8 th Inter. Software Process Workshop (ISPW8)*, 1993.

Biography

Giacomo Piccinelli is a researcher at the Hewlett-Packard Laboratories, UK. He received a B.Sc. in Computer Science (University of Pisa, Italy), and a M.Phil. in Information Engineering (Politecnico of Milan, Italy). His main research interests are Workflow Management Systems and Distributed Systems.