

Process Enactment in Virtual Software Organizations

John Noll
Computer Engineering Department
Santa Clara University
500 El Camino Real
Santa Clara, CA 95053-0566

Abstract

The conventional approach to process enactment employs a client-server architecture, in which a central engine executes process descriptions and often stores documents produced by the processes. This approach has been successful for automating processes in many situations. However, it relies on a significant centralized computing infrastructure, which does not fit distributed, loosely coupled “virtual organizations” such as open source software projects.

This paper presents a process enactment solution for such virtual organizations, based on three important features: A completely distributed, peer-to-peer architecture that eliminates the need for centralized computing and organizational infrastructure; process modeling based on independent process fragments that can be performed by a single individual actor; and, coordination entirely through products produced and consumed by concurrent processes.

Keywords: workflow, peer-to-peer, virtual organization, process enactment

1. Overview

The conventional approach to process enactment employs a client-server architecture, in which a central engine executes process descriptions and often stores documents produced by the processes [1]. Process participants (actors) interact with the engine through web browsers, whole environments, or task-specific tools, receiving guidance on what activities to perform, and how to perform them. This approach has been successful for automating processes in many situations, especially business processes in large enterprises.

In part, this is due to the engine’s global visibil-

ity of the organization: because all processes are enacted in a central location, concurrent processes can be closely coordinated, and work can be efficiently allocated to members of the organization.

However, this effectiveness comes with a cost: it requires significant central computing infrastructure, central administration, and a large support organization. These costs are readily borne by “enterprise class” organizations, but are prohibitive for other kinds of organizations that lack the physical and organizational structure required.

For example, open source software development can involve large numbers of people, yet the resources and infrastructure of the project are completely distributed among the participants. There is no building or IT organization to support a large computing facility.

This is an example of a “virtual organization”: a dynamic, loosely coupled, widely distributed group of actors who cooperate to achieve some common goal. Virtual organizations share the following characteristics:

- Complete distribution - members of the group are dispersed throughout the world, communicating primarily via email, the World Wide Web, Network News groups, and similar Internet-based mechanisms.
- Fluid membership - members join and leave the group at will.
- Organic structure - there is little hierarchy, and participants assume roles as needed, based on their perceived contribution toward the group’s goals.

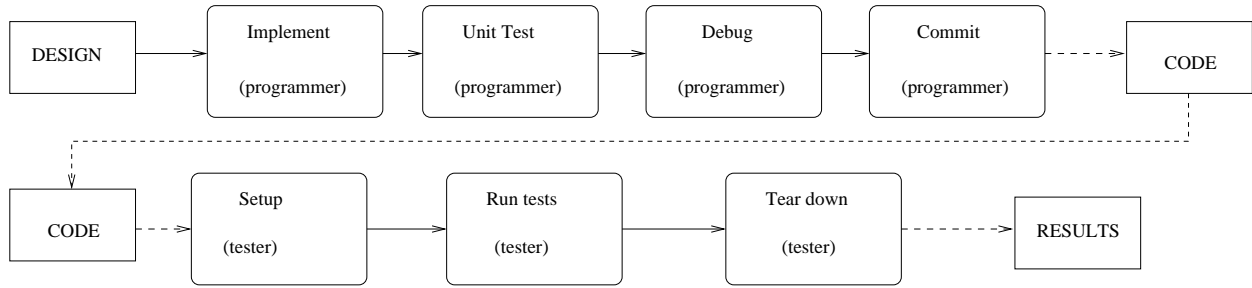


Figure 1. Coordinated Process Fragments

- Minimal infrastructure - a member’s employer may donate disk space for archiving the group’s documents, but there is no building or support organization, and little if any budget or paid staff.

Thus, while they could benefit from process support, these virtual organizations lack the structure and resources to deploy traditional enterprise-oriented client-server based process enactment.

In this paper, I propose a process enactment solution for virtual organizations, based on three important features: A completely distributed, peer-to-peer architecture that eliminates the need for centralized computing and organizational infrastructure; process modeling based on independent process *fragments* that can be performed by a single individual actor; and, coordination entirely through products produced and consumed by concurrent processes.

2 Approach

2.1 Modeling

Because the participants in a virtual organization are autonomous, distributed, and continually changing, processes have to be modeled differently than they would in a conventional enterprise approach. The process model must account for the fact that the process performers (actors) are independent, autonomous agents. This means they make decisions individually about what processes and tasks to perform, and when.

To accommodate this reality, my approach decomposes organizational processes into concur-

rent, independent process fragments, that cooperate to achieve a common goal. Each fragment consists of an ordered set of tasks (called *actions*) performed by a single actor; this feature means that a given actor can instantiate and perform a process fragment without affecting other actors, thus preserving each actor’s individual autonomy.

These processes are synchronized through the work products they share (see below), so that even though each actor works autonomously, their efforts contribute to the common organizational goal.

2.2 Coordination

In order to preserve the highest possible degree of autonomy and independence for actors, coordination between processes is modeled as resource sharing. Activities can block until a resource (product, document, or other artifact) is produced by another activity performed by a different actor. A predicate in the process description specifies the state that required resources must satisfy before the activity can proceed (the mechanism is described fully in [2]).

An example of a simple software implementation and test process is shown in Figure 1. This process is decomposed into two fragments: one bound to the programmer, comprising implementation tasks, and one bound to the tester, comprising system testing tasks. The two fragments are synchronized through the shared “CODE” resource; as a result, the testing fragment doesn’t begin until the programmer completes the Commit task, thus making the “CODE” resource available for testing.

Note that because the processes are indirectly coupled, it is not necessary for all activity to be

modeled, or enacted; enacted workflows can be coordinated with ad-hoc work or activities in another organization, through a shared resource. Thus, the implementation fragment can begin as soon as the “DESIGN” resource is available; but this resource can be produced by any process (or no formal process whatsoever).

2.3 Enactment

The enactment mechanism is based on a peer-to-peer architecture. Each actor runs an instance of the enactment system; these instances are entirely independent. It is assumed that the resources provided and required by actors’ processes are available from a shared distributed repository, such as the World Wide Web.

An example, showing enactment of the implementation-test process discussed above, is shown in Figure 2. In this example, one programmer and three testers coordinate to implement and test a body of code. They share resources through a global code repository (hosted by an entity such as SourceForge).

The arcs labeled “notify(code)” signify notification of a change in the code resource, an event detected by each of the participating testers’ enactment system.

3 Related Work

Several research efforts have addressed different aspects of the problem of providing process support for distributed, autonomous organizations.

For example, Exotica/FMDC [3] was system developed by IBM to support mobile and disconnected clients. Exotica/FMDC is an extension of the Exotica [4] workflow system, and is thus essentially a client-server architecture with features to allow the clients to operate when disconnected from the server.

Milos [5] specifically addresses process support for distributed software development teams. It is also a client-server architecture, but clients may be widely distributed across the Internet. The authors propose future extensions to the Milos architecture to allow Milos servers to coordinate with other Milos servers on a peer-to-peer basis. Thus, Milos

may evolve into a hybrid client-server/peer-to-peer architecture, in which work groups would share a server, and virtual organizations would consist of such workgroups assembled in a peer-to-peer fashion.

Magi [6] also follows a peer-to-peer architecture. Magi is a framework for constructing workflow applications, providing a set of protocols and services based on a scaled-down version of the popular Apache web server. Magi’s goal is to enable construction of workflow systems comprising mobile devices as well as traditional desktop computers. While Magi shares the goals of support for disconnected/mobile peers and wide distribution, it assumes that peers executing components of a process trust each other, so that they can create and update, as well as read, documents on peer nodes. Also, Magi assumes documents as well as processes are managed by the Magi servers. Thus, Magi could be viewed as a framework for deploying peer-to-peer workflow in an enterprise context.

4 Conclusion

The approach described herein has several distinctive features.

First, since individual actors each have their own enactment system, they can bring workflow support to groups to which they belong, without imposing on other group members.

Second, because the enactment system assumes that resources and products are stored in an external database, it works with, rather than replaces, existing data repositories. As a result, coordination can be achieved across organizational boundaries, and between workflow peers and ad-hoc processes, as long as the shared resources are accessible to both.

Further, because the coupling between coordinating actors is indirect, through a shared resource, there is no requirement for trust (or even awareness) between coordinating workflow peers. This enables extremely fluid, dynamic organizations in which participants can join at will without requiring administrative approval or action.

Finally, processes can be performed off-line, and synchronized periodically when desired or possible. This is especially important for mobile

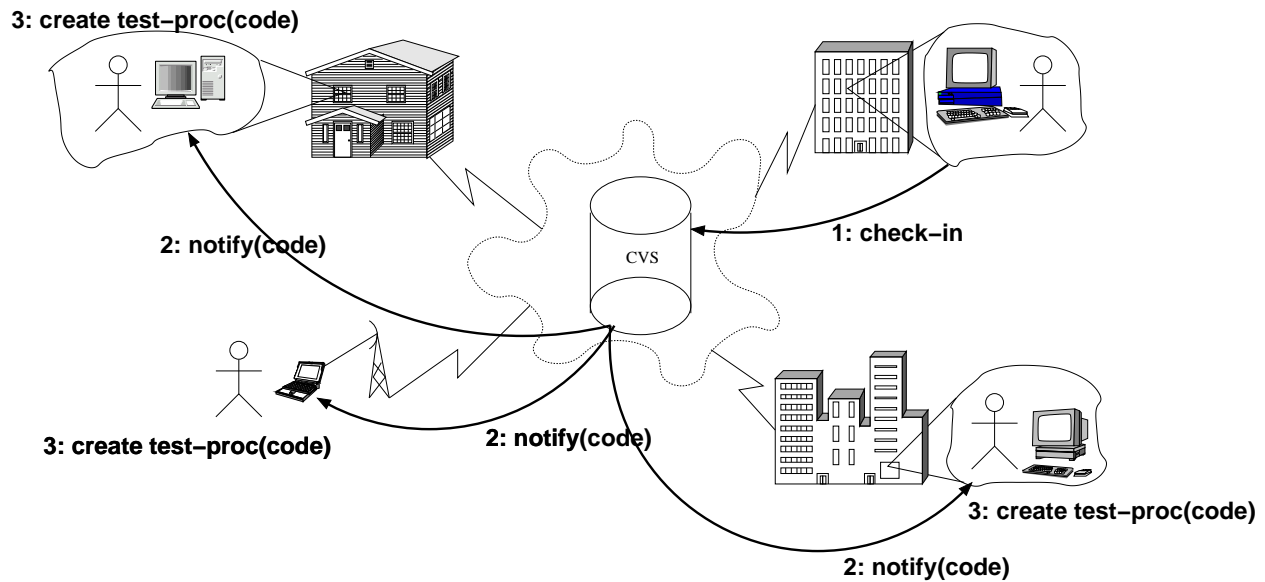


Figure 2. Peer-to-peer Enactment Architecture

actors, who often work with intermittent network connections.

References

- [1] David Hollingsworth. The workflow reference model. Technical Report TC00-1003, Workflow Management Coalition, January 1995.
- [2] Bryce Billinger. Modeling coordination as resource flow – an object-based approach. Master’s thesis, University of Colorado at Denver, September 2000.
- [3] Gustavo Alonso, Roger Gunthor, Mohan Kamath, Divyakant Agrawal, Amr El Abbadi, and C. Mohan. Exotica/FMDC: A workflow management system for mobile and disconnected clients. *Distributed and Parallel Databases, An International Journal*, 4(3):229–247, 1996.
- [4] C. Mohan, Gustavo Alonso, Roger Gunthor, and Mohan Kamath. Exotica: A research perspective on workflow management systems. *IEEE Data Engineering Bulletin*, 18(1):19–26, March 1995.
- [5] Frank Maurer, Barbara Dellen, Fawsy Bendeck, Sigrid Goldmann, Harald Holz, Boris Kotting, and Martin Schaaf. Merging project planning and web-enabled dynamic workflow technologies. *IEEE Internet Computing*, May/June, 2000.
- [6] Gregory Allen Bolcer. Magi: An architecture for mobile and disconnected workflow. *IEEE Internet Computing*, May/June, 2000.