Bulletin of the Technical Committee on

Data Engineering

March, 1995 Vol. 18 No. 1 IEEE Computer Society

Letters

Software Tools for a Process Handbook

Abraham Bernstein Chrysanthos Dellarocas Thomas. W. Malone
John Quimby
MIT-Sloan School of Management
Center for Coordination Science
1 Amherst Street
Cambridge, MA 02139
E-mail: {avi, dell, malone, quimby}@mit.edu

Abstract

This paper provides a progress report on the development of software tools in the Process Handbook project currently underway at the MIT Center for Coordination Science. We begin with a brief overview of the project as a whole. Then we focus on software tools emphasizing aspects that relate to workflow control. Finally, we conclude with a brief description of future avenues of research. The process handbook tools help (a) redesign *existing organizational processes, (b)*invent *new organizational processes that take advantage of information technology, and finally (c) automatically* generate software *to support organizational processes. An important related goal is the ability to (d)* import *and export process descriptions from and to other process modeling architectures. The approach combines in a novel way the ideas of process* decomposition*, process* specialization*, and the* coordination *of* dependencies *between activities. The paper presents an overview of findings from multiple implementations of this approach.*

1 The MIT Process Handbook Project

The goal of the Process Handbook project [8, 4] is to provide a firmer theoretical and empirical foundation for such tasks as enterprise modeling, enterprise integration [3], and process re-engineering [9, 6]. The project includes (1) *collecting* examples of how different organizations perform similar processes, and (2) *representing* these examples in an on-line "Process Handbook" that includes the relative advantages of the alternatives. The Process Handbook can be thought of as an Organizational-CAD tool. It is intended to help (a) *redesign* existing organizational processes, (b) *invent* new organizational processes that take advantage of information technology, and finally (c) *automatically generate software* to support organizational processes. An important related goal is the ability to (d) *import and export* process descriptions from and to other process modeling architectures.

The work in this project can be divided into three main categories, each of which is briefly discussed below: *theory development, field studies* and *tool building*.

1.1 Theory Development

A key element of the work is a novel approach to representing processes at various levels of abstraction. This approach uses ideas from computer science about inheritance and from coordination theory [10, 11] about managing dependencies. Its primary advantage is that it allows users to explicitly represent the similarities (and differences) among related processes and to easily find or generate sensible alternatives for how a given process could be performed. Section 2 provides a brief introduction to the major ideas behind our representation.

1.2 Field Studies

In parallel with developing theories and software to support the process handbook concepts, we are also conducting a series of field studies to gather data for inclusion in the handbook and to refine our concepts and methodologies for representing processes. Example studies we have conducted include:

- study of supply chain processes in athletic footwear companies
- study of software development management processes at a major computer manufacturer
- study of healthcare delivery processes at a community hospital
- comparative study of supply chain processes in (1) supplying pharmaceuticals in a hospital (2) telephone equipment installation in a regional telephone company, and (3) third-party software acquisition in a computer manufacturer.
- study of several processes in corporate banking

1.3 Tool building

In order to refine and test our ideas about how a process handbook should work, we have developed a series of prototype systems embodying various aspects of the functionality envisioned in our proposal. Sections 3 and 4 provide a detailed review of where we are and what our vision is.

2 Theoretical Background

A key to this project is developing novel techniques for representing organizational processes. Our goal is to use advanced process representation techniques in order to:

- assist process analysis and identification of inefficiencies
- facilitate generation and comparative evaluation of alternative processes

Space precludes a full description here of our approach to representing processes. Briefly, however, the representation used in the handbook combines three basic concepts in a novel way to create a taxonomy of processes (see [8] for a complete description).

1. *Decomposition.* Processes are decomposed into activities, which may in turn be further decomposed into subactivities. Decomposition allows the nesting of processes within processes, and allows the handbook to share and re-use process descriptions throughout the taxonomy. Decomposition itself is, of course, not a novel element in process representation, but it is combined here in new ways with the other two elements.

Dependency	*Examples of coordination processes for managing dependency*
Shared resources	"First come/first serve", priority order, budgets, managerial decision, market-like bidding
Task assignments	(same as for "Shared resources")
Flows	
Prerequisite constraints	Notification, sequencing, tracking
Inventory	Inventory management (e.g., "Just In Time", "Economic Order Quantity")
Usability	Standardization, ask users, participatory design
Design for manufacturability	Concurrent engineering
Simultaneity constraints	Scheduling, synchronization
Task / subtask	Goal selection, task decomposition

Figure 1: Examples of common dependencies between activities and alternative coordination processes for managing them. (Indentations in left column indicate more specialized or decomposed versions of general dependency types.)

2. *Specialization.* Processes (and activities) are specialized in a manner similar to a traditional type hierarchy. Unlike a simple object hierarchy however, each node is itself a complex entity that inherits a decomposition and the associated dependencies from its parents. Loosely speaking, our notion of process specialization can be thought of as a "dual" of traditional object-oriented programming. Instead of inheriting down a hierarchy of objects with associated actions (methods), we inherit down a hierarchy of actions (processes) with associated objects.

3. *Dependencies.* The third key concept is the notion from coordination theory that coordination processes can be thought of as ways of managing dependencies among activities [10, 8]. Organizational processes can be viewed as containing both *production* and *coordination* components. The production component includes the process activities that are physically or logically necessary to achieve the stated goals of the process. The coordination component consists of the activities necessary to properly manage the dependencies among production activities.

We are testing the hypothesis that most coordination processes encountered in practice can be viewed as alternative ways of managing a relatively small set of elementary dependency types. From this perspective, we can build a taxonomy of basic dependency types and associate each dependency type with a specialization hierarchy of alternative coordination processes for managing it. Furthermore, it seems that many coordination processes are applicable across a large range of functional domains. For example, a coordination process used to manage a flow dependency in a manufacturing process, may be used intact or with minor modifications to manage a similar dependency in a finance process.

Table 1 [8] suggests the beginnings of such an analysis. The table shows a set of common dependencies between activities together with examples of alternative coordination processes used to manage them.

Note that dependency types themselves form a specialization hierarchy, with more specific types inheriting (and possibly specializing) the set of managing processes of more general dependency types. For instance, *task assignment* can be seen as a special case of allocating shared resources. In this case, the "resource" being allocated is the time of people who can do the tasks. This implies that the coordination processes for allocating resources in general can be specialized to apply to task assignment.

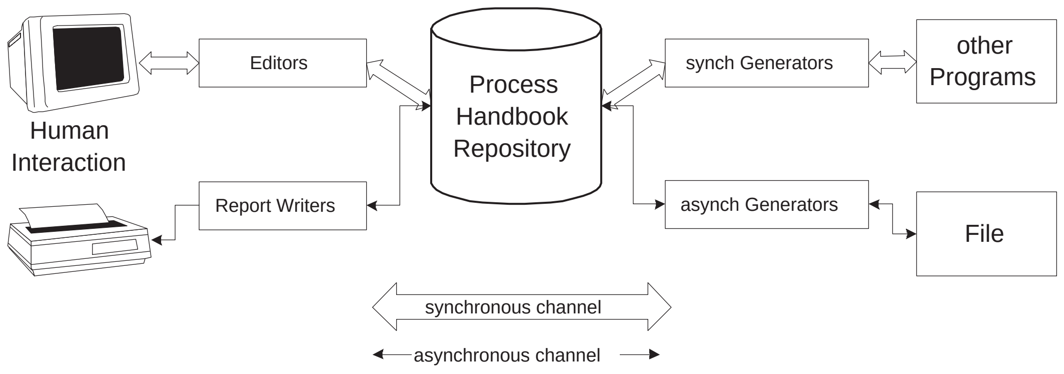

Figure 2: The Process Handbook Architecture

2.1 Advantages of activity-dependency representation

Abstraction enhances understanding. Coordination activities are often relatively "low-level" (e.g., "Complete requisition form", "Ship packet by courier", etc.), compared to the production activities whose dependency they manage. Traditional process representations, which mix production and coordination activities in the same picture, very fast become cluttered with detail. This hinders understanding the essence of the process, as well as the purpose of coordination activities. Separating coordination activities and "hiding" them behind dependencies, results in simpler representations, which highlight the essence of a process. In addition, it associates coordination activities with their underlying dependencies. Thus, the purpose (or lack of it) for each coordination activity is made clear.

Dependencies enhance generativity. As we mentioned earlier, each dependency type is connected to a specialization hierarchy of alternative coordination processes for managing it. The association of an existing coordination process with its underlying dependency, not only improves understanding of the business process, but also provides immediate access to a large number of alternative ways of performing that coordination, extracted from business processes in many different domains. This helps produce ideas for generating improved alternatives for the studied process.

3 The Process Handbook Tools

3.1 The Process Handbook Architecture

The Process Handbook usage scenarios propose several different types of usage by humans and machines. In general, there are two types of interaction with the process handbook repository: *synchronous* and *asynchronous*.

The synchronous channel tools are directly connected to the process handbook repository throughout their usage. In most cases they change the process handbook repository. We call our synchronous tools that enable human computer interaction *editors*. Synchronous tools that enable inter program communication are synchronous generators. Synchronous channels usually use function libraries to access the process handbook repository API directly in combination with other standard inter process communication protocols like sockets, DDE, RPC, and OLE.

Report writers are asynchronous tools that interact with people through some other medium. Asynchronous generators extract certain parts of the process handbook repository and generate some output file.

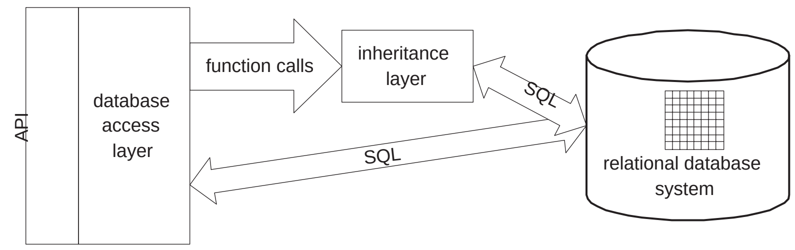

Figure 3: The Repository Architecture

3.2 The Repository

The process handbook repository is the "memory" of the process handbook. It stores the process descriptions using the representation techniques described in section 2 and ensures the data integrity. The span of the usage scenarios asks for a scaleable solution that can be used on different systems ranging from laptops to large server based process repositories. To solve this scalability problem we chose a relational database system as our storage engine, which is wrapped by two layers that simplify the interaction of the different tools with the repository.

The inheritance layer provides the process inheritance functionality needed by the process handbook. For instance, it propagates changes made to a process at one level to all the specializations of that process at lower levels. It would presumably have been easier to use an object-oriented database system for this purpose. However we were unable to find a suitable object-oriented database available for the PC/Windows environment in which our system is implemented.

Our database access layer [2] encapsulates all database-operations. This middleware-layer introduces an abstraction layer between the object-oriented data and the relational database system. Through its API, it offers different means of navigation through the object- like Process Handbook space. This abstraction layer is also being designed to integrate an additional repository based on Lotus Notes. This alternate repository is restricted in the sense that it does not have the full richness of the relational database-structure. However, it is optimized to store rich text descriptions of processes, doing full text retrieval, and support the collaborative dialogues of field researchers working remotely in the field. One of our goals for the abstraction layer (not yet fully implemented), is that all programs accessing the Process Handbook repository can access the information in both storage engines.

3.3 The Editors

One of the major aspects of the process handbook project is to help its users in their task of browsing through process descriptions, changing them, simulating, and executing them. The way users interact bi-directionally with the process handbook repository is through Editors. To test the hypothesis that the process handbook captures a superset of the information in other systems, a number of editors for different types of notations have been implemented.

The first group of editors are tree editors. They are used to browse and edit the specialization and decomposition trees of processes. They are implemented as graphical, direct manipulation programs, and therefore provide very direct access to structural information in the process handbook repository.

The second type of editors are additional (non tree-) graphical editors. They represent process information in different notational formats. These editors so far include an IDEF0 editor, a Dataflow editor, and a Flow-chart like editor that allows the exact representation of dependencies between activities. This latter editor also supports users in choosing the appropriate managing activities for dependency and can be used to activate a special software generator (see section 3.5)

The third group of editors are text based. They allow navigation through the detail sheets of processes, and dependencies with hypertext-like links. These editors are mainly used to enrich the process handbook repository with detailed information about its objects rather than to change structural information. These details include full text descriptions and statistical information about the processes which can be used to drive a process animation viewer, and can also be exported for use by external applications. An example of such an external application would be a commercial process simulator.

3.4 The Report Writers

Report writers are used to produce paper representations of the information stored in the process handbook repository. This includes printing the graphical notations, printing textual information about processes and exporting statistical data. In some cases, data is exported to a specialized drawing program for users to further edit manually.

3.5 Synchronous Generators

A whole set of programs focuses on direct communication between the process handbook repository and outside applications. We call these tools synchronous generators, because they are in bi-directional communication with the repository.

One of the tools with which we are experimenting takes process handbook process descriptions and executes it as open nested transactions [12]. This particular program extracts the process descriptions from the process handbook repository and acts as coordination process for those processes. By observing the rules of open nested transaction execution the system ensures the correct execution of the processes. The current prototype based on [1], communicates with independent interfaces, supporting actions in a distributed environment. In addition to ensuring the execution of the transactional workflows, the system has a management interface that graphically shows the current state of a particular, focused activity.

3.6 Asynchronous Generators

In addition to direct communication with other programs, it is also possible to communicate indirectly with other programs through asynchronous file exchange. One of the efforts related to the process handbook project has been the development of a Process Interchange Format (PIF see [7]). The PIF was developed by a working group consisting of representatives from several universities and companies. The goal of the PIF project is to automatically exchange process descriptions between different systems.

To achieve this goal, each system will only need to have a translator to convert data between its native format and the common PIF format. The format itself is a representation of process entities, such as activities and resources, along with their attributes and relationships. In addition to the core set of object types, the PIF provides a framework for extending this set of types to include additional information needed in specific applications. Currently, our PIF translators are the primary method of communication for handbook tools operating on different databases or platforms. In the larger scope of the PIF Working Group, we have also performed experimental translations among systems developed by different research groups. Our team is writing a platform-independent C++ toolkit [5] which will simplify the task of writing future translators, not only for our group but also for anyone who would like to share business process descriptions using this format.

4 Future Vision

The wide range of user profiles of the Process Handbook from business school students to organizational theorists to management consultants presents an interesting set of user interface challenges. Advancing the usability of the Process Handbook based on a wide range of usage scenarios is one line of ongoing work. This includes more advanced process representations and richer manipulation metaphors that can be tailored according to usage patterns of the tool users. Even though we are encouraged by our progress so far in process representation, we believe that process representation is still in its infancy. By analogy to the history of musical notation, our current work can be seen as analogous to the use of a G Clef. Imagine what it will be like to add measure bars, time signatures, and dynamic markings!

A particularly rich area for ongoing research is the automatic and semi-automatic generation of operational systems and the control of those systems from the Process Handbook. Can the imagining, specification, implementation, deployment, operational control, and evolution of a production system be supported from within a single integrated framework? What are the changes in tool sets and in the design practices required to support such a vision? These are the kinds of questions our work so far on the Process Handbook raise for us.

5 Acknowledgments

The Process Handbook is a team project. A number of researchers are working on and have made significant contributions in the various areas of the project. We would especially like to acknowledge the work of Erphan Ahmed, Steven Buckley, Frank Chan, Kevin Crowston, Naved Khan, Jintae Lee, Greg Pal, Brian Pentland, Narasihma Rao, George Wyner, Greg Yost, and Gilad Zlotkin.

References

[1] Bernstein A. Specification and implementation of workflows in banking environments. Master's thesis, Diploma Thesis at the Computer Science Division of the Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1993.

[2] Erfanuddin Ahmed. Thesis for master of engineering degree in electrical engineering and computer science. Master's thesis, Massachusetts Institute of Technology, June 1995.

[3] Petrie C. Enterprise integration modeling. 1992.

[4] T. W. Malone K. Crowston C. Dellarocas, J. Lee and B. Pentland. Using a process handbook to design organizational processes. *Proceedings of the AAAI Spring Symposium on Computational Organ. Design*, pages 50–56, 1994,.

[5] Frank Yeean Chan. The round trip problem: A solution for the process handbook. thesis for master of engineering degree in electrical engineering and computer science. Master's thesis, Massachusetts Institute ofTechnology, June 1995.

[6] Champy J. Hammer M. *Reengineering the Corporation.* Harper Business, 1993.

[7] Yost G. Lee J. and the PIF Working Group. The pif process interchange format and framework, version 1.0. 1994.

[8] Lee J. Malone T.W., Crowston K. and Pentland B. Tools for inventing organizations: Toward a handbook of organizational processes. *Proceedings of the 2nd IEEE Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises*, 1993.

[9] Davenport T.H. *Process Innovation.* Harvard Business School Press, Boston, Massachusetts, 1992.

[10] Malone T.W. and Crowston K.G. Toward an interdisciplinary theory of coordination. 1991.

[11] Malone T.W. and Crowston K.G. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26, 1994.

[12] Schaad W. Weikum G., Deacon A. and Scheck H. Open nested transactions in federated database systems. *IEE Data Engineering No. 2, Special Issue on Workflow and Extended Transaction Systems*, 16, 1993.