THE PRODUCT WORKBENCH: AN ENVIRONMENT FOR THE MASS CUSTOMIZATION OF PRODUCTION-PROCESSES

TO APPEAR IN
THE PROCEEDINGS OF WITS '98,
HELSINKI, FINLAND

ABRAHAM BERNSTEIN

Address
Massachusetts Institute of Technology
Center for Coordination Science
Room E40-179
1 Amherst Street
Cambridge, MA 02139, U.S.A.
Phone: +1 (617) 253 8410
Fax: +1 (617) 253 4424
Email: avi@mit.edu

Keywords
Automatic Program Generation, Business Process Redesign, CASE, Process innovation

The Product Workbench: An Environment for the Mass-Customization of Production-Processes

ABSTRACT

This article investigates how to support process enactment in highly flexible organizations. First it develops the requirements for such a support system. Then it proposes a prototype implementation, which offers its users the equivalent of a CAD/CAM-like tool for designing and supporting business processes. The tool enables end-users to take flexible building blocks of a production process, reassemble them to fit the specific needs of a particular case and finally export its description to process support systems like workflow management systems.

INTRODUCTION: IT IN AN ECONOMY OF PERPETUAL CHANGE

A variety of organizational observers (Argyris and Schön (1996), Boyton, Victor and Pine (1993), Laubacher, Malone and MIT-Scenario-Working-Group (1997) and Nonaka and Takeuchi (1995) among others) predict new organizational forms, which are presumed to be highly flexible, continuously changing their form, their product range and their structure. Firms will evolve into new, more flexible forms in which the interrelations between the organizational units are not organized by a hierarchical information flow but much more by a network of communicative links (see Van Alstyne (1997)). How can we support the enactment of highly flexible processes in such an organization?

While rapid prototyping-environments and CASE-tools have been addressing the problems of continuous change they usually produce solutions which are either not scaleable, require highly specialized knowledge (especially with CASE-tools) or are limited to a single, proprietary enactment environment (like a workflow-system or a transaction monitor). This paper reports on the implementation of a prototype system to support the rapid development of new production processes by end-users, which can then be enacted on a variety of execution platforms. Building on ideas from the product development and innovation literature, it combines concepts from rapid-prototyping, component based programming, object-oriented programming, knowledge-based systems and human-computer interface design to develop a product workbench for business users.

The remainder of the paper is structured as follows. First we analyze the requirements for a product workbench using foundations from the literature on product innovation, knowledge-transfer and artificial intelligence. Second we describe a prototype implementation, which will be illustrated using a practical scenario from the financial services industry. Finally, we evaluate the proposed solution and discuss future work.

Analysis of the Requirements and Theoretical Foundations

Boyton, Victor and Pine (1993) build a framework to explain new production paradigms. In this framework they analyze production as varying on two dimensions: product change and process change. In the mass-production setting, a product and its production process are highly stable. In a research department both product and production process are highly dynamic leading to high costs and low volumes. The question then arises as to whether it is possible to reduce the costs and sell high volumes of customized products. The idea of *mass-customization*[1], in which the product changes to fit specific demand and the production is organized around '*…loosely coupled networks of modular, flexible processing units…*[2]' seems to allow such a production scheme. One of the scenarios by Laubacher, Malone and MIT-Scenario-Working-Group (1997) proposes a similar structure for future organizations. A network of loosely coupled specialists (in most cases one person firms), who come together to produce a highly customized product (of batch size one) and then reconfigure to meet the challenges of the next project. To support such an organization, an IT-support system will thus have to *enable people to take flexible building blocks of a production process and reassemble them to fit the specific needs of a particular case.*

1 Gilmore and Pine (1997) offer a more detailed analysis of mass customization.

2 Boyton, Victor and Pine (1993), p. 49

Unfortunately end users (e.g. account managers in a bank) are usually not trained to reconfigure and reassemble existing processes, a job which is usually performed by business analysts. We therefore need to *'unstick' the process design knowledge* (von Hippel (1996)) and make it accessible to end-users by encoding it in component-like building blocks and consistency rules of a design environment. The result would be a type of integrated *CAD/CAM*[3] *tool for business processes*. This is consistent with von Hippel's (1996) observations in the ASIC's and computer telephony industry.

The component-based approach contains the problem of how to organize the large number of components in order to make them accessible. Experience in AI has shown that it often makes sense to construct some type of taxonomy of components in which similar components can be found close together, leading to the development of frame inheritance networks and object type hierarchies[4] (Brachman and Schmolze (1985)). Furthermore the usage of template (or prototype) hierarchies, a form of simplified frame inheritance networks, has been observed to be useful in settings with end-user development (MacLean et al. (1990)). Thus a *template-oriented component-hierarchy*, which can also hold previously completed cases as templates, seems to be advantageous in helping to solve our problem.

[3] **C**omputer **A**ided **D**esign/**C**omputer **A**ided **M**anufacturing

[4] Borning and O'Shea (1987) find that object-oriented concepts can be difficult to understand. We therefore chose to use the slightly simpler notion of template hierarchies.

The Implementation

The Basis: Process Handbook

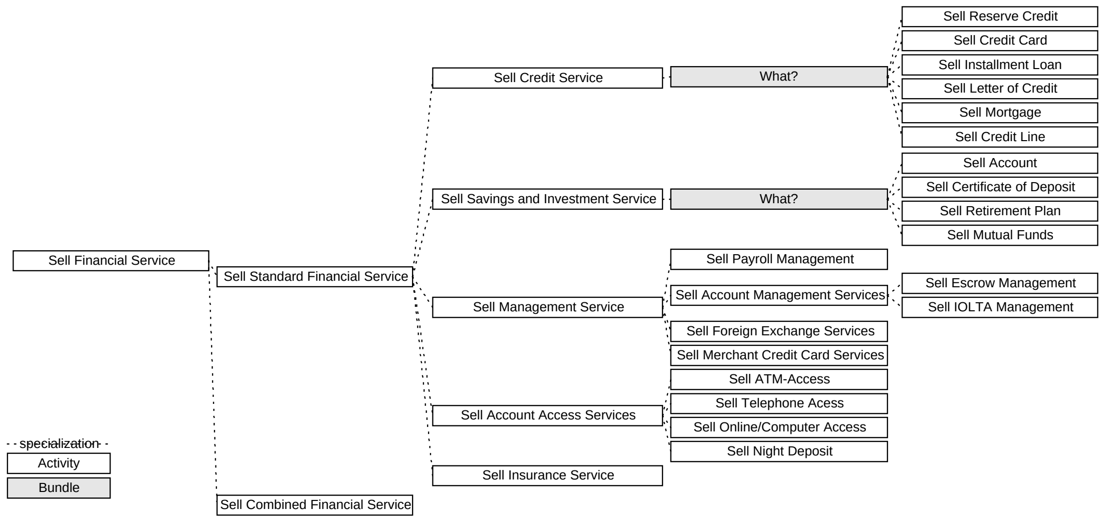
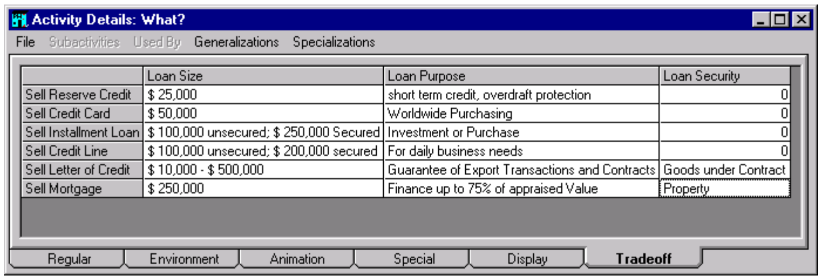


Figure 1: Specialization Hierarchy for 'Sell Financial Service' (based on BankBoston (1998))

The architectural basis of the implementation is the Process Handbook process knowledge base (see Malone et al. (1997) and Bernstein et al. (1995)). The goal of the process handbook project at MIT, which has been under way for over six years, is to develop a process repository and associated tools to allow users to quickly retrieve and effectively exploit the process knowledge relevant to their current challenge. Two of the process handbook's features are central to our endeavor: process inheritance and the distinction of processes and their interdependencies. We will therefore explain them before we go on to other parts of the implementation.

Process specialization takes features of frame inheritance networks (Brachman and Schmolze (1985)) and transfers them into the process domain. It arranges processes in a hierarchy of 'types of' or 'ways of' doing things which goes from very generic processes at one end to very specialized processes at the other end (see Figure 1). This specialization hierarchy offers the capabilities we need to store cases, templates and thus process components. We can use the more generalized processes as templates and specialize them as we develop a new product. Past cases would thus usually be leaves in the specialization hierarchy, which could also be used as templates for new products. At some levels the hierarchy even has special objects (called bundles), whose role it is to facilitate the classification of the specializations of a process by offering a specific dimension by which the processes are compared (see Figure 2).



	Loan Size	Loan Purpose	Loan Security
Sell Reserve Credit	$ 25,000	short term credit, overdraft protection	0
Sell Credit Card	$ 50,000	Worldwide Purchasing	0
Sell Installment Loan	$ 100,000 unsecured; $ 250,000 Secured	Investment or Purchase	0
Sell Credit Line	$ 100,000 unsecured; $ 200,000 secured	For daily business needs	0
Sell Letter of Credit	$ 10,000 - $ 500,000	Guarantee of Export Transactions and Contracts	Goods under Contract
Sell Mortgage	$ 250,000	Finance up to 75% of appraised Value	Property

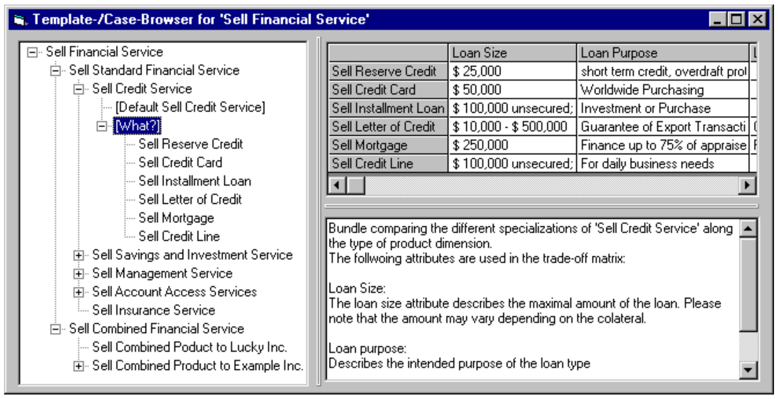
Figure 2: Trade-off matrix showing the alternative specializations of 'Sell Credit Service' compared by 'Loan Size', 'Loan Purpose' and 'Loan Security'

Due to its roots in coordination science (see Malone and Crowston (1994)) the Process Handbook *distinguishes dependencies from their coordination mechanisms*. Dependencies represent the flow of physical resources (e.g. trucks) or informational resources (e.g. signals) between two activities. Alternatively they can also represent the sharing of such a resource (e.g. a meeting room), the fit thereof (e.g. two artists cooperating in the writing of a song) or some combination of the types presented. Coordination processes are the activities that manage those dependencies. In the case of a flow dependency for example, one has to make sure that the resource is transported from the producer to the consumer. This perspective can be extremely useful for solving our problem, because we can hide all the coordination mechanisms from the user of the product workbench, where she doesn't need to know about it and thus reduce the complexity of the product assembly task. In some instances where the user of the product design workbench is particularly interested in issues of coordination (e.g. when she is the general contractor of a building project) she will want to highlight coordination problems.

The Scenario

We will now introduce the Product Workbench by using a usage scenario that illustrates how an account manager in a bank could use it to construct a new financial product.

The account manager in a commercial bank represents the customer's single point of access. Let us assume that a customer wants a special revolving loan, which is coupled with an investment fund. To be more specific: the customer wants an account, which automatically adjusts its structure depending on the amount of money in it. If the account has a positive balance, then the sum should be invested in a money market fund. In the case of an overdraft situation, the money should be automatically drawn from the revolving loan (or from the money market fund if available). This setting resembles a complex checking account with overdraft protection and an active investment of the funds as opposed to a fixed low interest.



Template-/Case-Browser for 'Sell Financial Service'

- Sell Financial Service
  - Sell Standard Financial Service
    - Sell Credit Service
      - [Default Sell Credit Service]
      - [What?]
        - Sell Reserve Credit
        - Sell Credit Card
        - Sell Installment Loan
        - Sell Letter of Credit
        - Sell Mortgage
        - Sell Credit Line
    - Sell Savings and Investment Service
    - Sell Management Service
    - Sell Account Access Services
    - Sell Insurance Service
  - Sell Combined Financial Service
    - Sell Combined Poduct to Lucky Inc.
    - Sell Combined Product to Example Inc.

	Loan Size	Loan Purpose	L
Sell Reserve Credit	$ 25,000	short term credit, overdraft prol	
Sell Credit Card	$ 50,000	Worldwide Purchasing	
Sell Installment Loan	$ 100,000 unsecured;	Investment or Purchase	
Sell Letter of Credit	$ 10,000 - $ 500,000	Guarantee of Export Transacti	(
Sell Mortgage	$ 250,000	Finance up to 75% of appraise	F
Sell Credit Line	$ 100,000 unsecured;	For daily business needs	

Bundle comparing the different specializations of 'Sell Credit Service' along the type of product dimension.
The follwoing attributes are used in the trade-off matrix:

Loan Size:
The loan size attribute describes the maximal amount of the loan. Please note that the amount may vary depending on the colateral.

Loan purpose:
Describes the intended purpose of the loan type

Figure 3: The template/case-browser

In the classical banking world this request would be a disaster. It would involve the implementation of a number of features in the bank's accounting systems: a time consuming project. Our account manager on the other hand knows that the general building blocks for such a request are in her product workbench. She first starts up her *template/case browser* (see Figure 3, and Figure 6, lower left) in order to find an appropriate template for the requested product. The template/case browser offers a three pane (frame) view. On the left side it displays a hierarchical grouping of the possible choices. When one of those choices is selected the right side of the browser shows some additional information about the chosen element. At the bottom of the right side is a detailed description of the item and at the top is a comparison matrix (like in Figure 2) of the possible choices. This browser thus allows her to navigate through the process knowledge base specialization hierarchy stored in the process handbook and make decisions about the appropriateness of processes by (1) offering detailed information about the process and (2) comparing the different specializations. She can choose either a generic process or a product constructed for another customer (a previous case) as a template for the new product. In our case she chooses 'Sell Combined Financial Product' as a template and calls the process 'Sell Combined Product to Lucky Inc.'.

The *integrity checker* then takes the chosen process template and tests whether it is in an enactable format (comparable to the first pass of a two-pass compiler). First it replaces all dependencies (not shown) with their specified managing process. Second it examines all processes using a depth-first algorithm on the process decomposition tree[5]. When encountering a leaf process it checks whether all necessary references (e.g. to an executable program and an actor) are well defined. Nodes are tested as soon as all their sub-activities are examined by scrutinizing the connections between its sub-processes. Finally the integrity checker points out failure of those tests by directing the user to the problems. This is achieved by opening up the appropriate browser (decomposition browser for processes, dependency browser for dependencies) and highlighting the problem areas (see Figure 5)[6]. By examining the problem areas with the case/template browser as described above, the account manager will be able to find well-specified processes for the problematic processes, to further refine the process design and then to reinitiate the integrity checker (see Figure 4).

[5] Dellarocas (1996) describes in detail a similar algorithm operating on a comparable data-structure.

[6] Figure 5 shows the result of the integrity checker if it would be run after step 1 in Figure 4

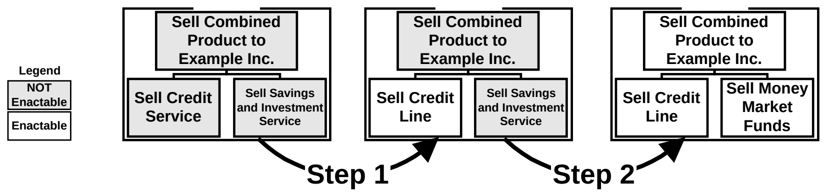

Legend
NOT Enactable
Enactable
Sell Combined Product to Example Inc.
Sell Credit Service
Sell Savings and Investment Service
Sell Combined Product to Example Inc.
Sell Credit Line
Sell Savings and Investment Service
Sell Combined Product to Example Inc.
Sell Credit Line
Sell Money Market Funds
Step 1
Step 2

Figure 4: Incremental and iterative refinement of the process 'Sell Combined Product to Example Inc.'

In our case the next stage is to examine the problem areas as pointed out by the integrity checker in a decomposition browser (see Figure 5), which offers a tree-like view, in order to determine which parts have to be replaced with other components. Using the template/case browser she will browse the specialization hierarchy of the non-determined processes, e.g. 'Sell Credit Service', and then replace each such process with one of its well-defined specializations, e.g. 'Sell Credit Line' (see also Figure 4, Step 1). After one more replacement (Step 2 in Figure 4) she can reinitiate the integrity checker. This leads to the incremental refinement of the process by replacing all the under-defined components with well-defined ones.

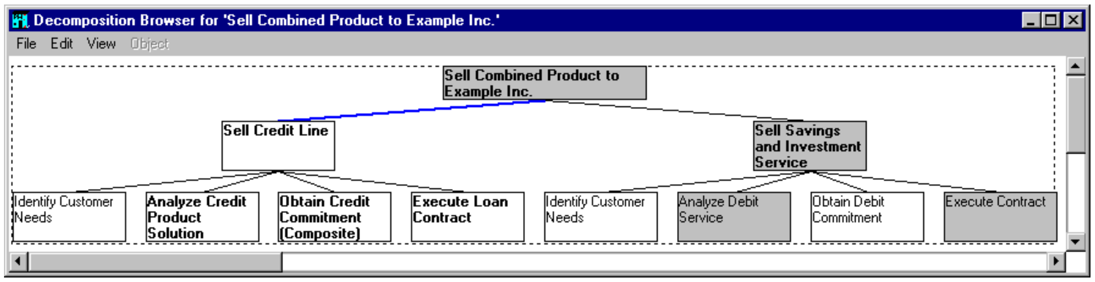

Decomposition Browser for 'Sell Combined Product to Example Inc.'
File Edit View Object
Sell Combined Product to Example Inc.
Sell Credit Line
Sell Savings and Investment Service
Identify Customer Needs
Analyze Credit Product Solution
Obtain Credit Commitment (Composite)
Execute Loan Contract
Identify Customer Needs
Analyze Debit Service
Obtain Debit Commitment
Execute Contract

Figure 5: Integrity Checker pointing out problems in the decomposition browser by coloring the processes 'Analyze Debit Service' and 'Execute Contract' in a darker color. 'Sell Savings and Investment Services' and 'Sell Combined Product to Example Inc.' are also colored dark because they contain non-enactable sub-processes.

Finally, when the integrity checker finds no problems in the process description it passes it to the *code generator*, which traverses the process description and generates the appropriate scripts and programs. To surpass the limitation given by a single process support system, the product workbench can generate scripts or programs for multiple platforms, which interrelate as defined in the process map. For example the process could be partly enacted on an ERP and partly on a transaction-processing host, which are coordinated by a workflow-management system (WFMS).

Currently the code generator supports the commercial WFMS Staffware™ and an agent based research WFMS[7]. At last the code generator contacts the involved systems and ensures that the scripts and programs are installed and ready for execution. The account manager has accomplished the task of designing a new customized product and could start the process to service her customer.

So far all dependencies have been hidden from the account manager. She will never have to deal with dependencies, provided that the interface of the under-defined process-place holders and the determined replacing components (i.e. well-defined processes) are compatible. When there is a problem with dependencies (like the absence of a coordination mechanism), then the integrity checker will point those out in the *dependency editor*, which offers a flow-chart like view of the process and its dependencies (see Ahmed (1998)). Using the case/template browser the account manager can then further refine her product design and by replacing a non-determined dependency with one of its well-defined specializations.

The overall architecture of the Product Workbench, which supports this scenario, is summarized in Figure 6.

[7] The only task necessary for the generation of scripts/programs for additional enactment support systems is the programming of the code generator for the targeted platform.

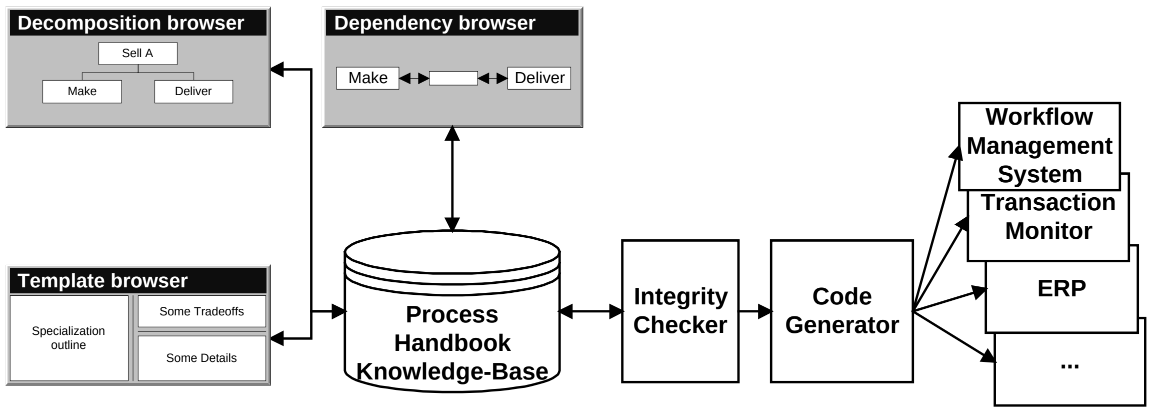


Figure 6: Overall product workbench architecture

Discussion

Evaluation of the Solution

The proposed solution fulfills the requirements developed in the analysis section above: It reduces the knowledge-transfer problem by ‘unsticking’ the process design knowledge and providing high-level process-based operations, understandable to an end-user. Furthermore it offers a repository of available high-level building blocks, which are structured in a non-specialist accessible fashion (in our implementation a template hierarchy). It thus enables end-users to *take flexible building blocks from the process handbook database and flexibly reassemble them* according to the needs of a particular customer. We therefore think that it supports the rapid incremental development and mass-customization of production processes. We also believe that it could consequently support the enactment of processes in highly flexible organizations.

In some cases a production process may require strictly transactional behavior in one part of its enactment, which can be supported by a transaction monitor. But in another part it may also rely on a loosely coupled succession of activities, which are best supported by a group-ware discussion database as a coordination mechanism. Therefore we believe that our system’s ability to export to multiple enactment support environments will make it more suitable for the support of mass-customization than workflow-management systems (WFMS), which usually only support their own system as enactment support.

Furthermore our system proposes to close the gap between high-level concepts and low-level program-code generation by focusing on business processes and an inheritance framework of components, which offer a better abstraction than traditional CASE-tools. Therefore we believe that the system will be usable by end-users and not only by specialists.

The product workbench does however forgo some of the flexibility of CASE-systems and WFMS by using a component based-approach, in which the end-users can only assemble their production processes out of existing components. Developing good and useful product components will be a key success-factor for such a system, which cannot be accomplished by domain specialists alone, but will have to involve information systems specialists in order to integrate the components with the back-end systems. Furthermore the system will face the usual challenges of component-based systems (e.g. integration problems with the transactional behavior of a collection of components[8]).

Future Work

There are a variety of open questions in connection with the product workbench. The next step will be to compose a library of real-world components. These, and an integration of the product workbench into a standard corporate work-environment, could be used to explore the practicality of the tool in a real world setting.

A parallel avenue of investigation explores alternate uses of the enactment scripts. One could, for example, use the script in order to simulate its enactment, which would help to accurately estimate its cost and then price it. This estimate could be improved by connecting the simulation engine to real-world pricing and scheduling information about internal and external resources involved in the production process. Thus an account manager could quote the price and a planned

[8] Such problems can arise, for example when dependencies between components are not sufficiently declared and a production process 'deadlocks' itself. While there are extended transaction mechanisms to deal with complex nested transaction schemes (see Elmagarid (1992)), component developers will have to document all potential side effects (i.e. dependencies to other resources and activities) of their components to ensure the correct application of those mechanisms.

delivery date (using the scheduling information) for a mass-customized product before the firm would have to invest in the enactment of its production.

CONCLUSION

In this paper we have described a system called the Product Workbench. We believe that its component-based design approach paired with its template-oriented repository of components shows how systems can enable end-users to mass-customize production processes. Furthermore we believe that this approach is likely to be especially suited to supporting the novel organizational structures of the future.

ACKNOWLEDGMENTS

We'd like to thank the members of MIT - Center for Coordination Science, especially Thomas W. Malone for his advice and Zia Ahmed for his fast responses to my implementation requests, for their invaluable contributions to the ideas underlying this paper.

REFERENCES

Ahmed, Zia. 1998. "An Integrated Dependency Editor for the Process Handbook." in *Department of Electrical Engineering and Computer Science*.M. Eng. Thesis. Cambridge, MA: Massachusetts Institute of Technology.

Argyris, Chris, and Donals A. Schön. 1996. *Organizational Learning II: Theory, Method, Practice*. Reading, MA: Addison-Wesley.

BankBoston. 1998. "Focus On Making Your Business A Success: Business Focus Banking." .Sales Catalog. Boston, MA: Bank Boston.

Bernstein, Abraham, Chrysantos Dellarocas, Thomas W. Malone, and John Quimby. 1995. "Software Tools for a Process Handbook." *IEEE-Data Engineering* 18 (1):41-48.

Borning, A., and T. O'Shea. 1987. "An Empirically and Aesthetically Motivated Simplification of Smalltalk-80." Pp. pp. 155-165 in *European Conference on Object-Oriented Programming*Paris.

Boyton, Andrew C., Bart Victor, and B. Joseph II Pine. 1993. "New Competitive Strategies: Challenges to Organizations and Information Technology." *IBM Systems Journal* 32 (1):60-64.

Brachman, R. J., and J. G. Schmolze. 1985. "An Overview of the KL-ONE Knowledge Representation System." *Cognitive Science* 9 p. 171-216.

Dellarocas, Chrysanthos. 1996. "A coordination Perspective on Software Architecture: Towards a design Handbook for Integrating Software Components." Pp. 288 in *Department of Electrical Engineering and Computer Science*.Ph.D. Thesis. Cambridge, MA: Massachusetts Institute of Technology.

Elmagarid, A. K. 1992. *Database Transaction Models for Advanced Applications*: Morgan Kaufman.

Gilmore, James H., and B. Jospeh II Pine. 1997. "The Four Faces of Mass Customization." *Harvard Business Review* (January-February):91-101.

Laubacher, Robert J., Thomas W. Malone, and The MIT-Scenario-Working-Group. 1997. "Two Scenarios for 21st Century Organizations: Shifting Networks of Small Firms or All-Encompassing 'Virtual Countries'?" .21 Century Initiative Working Paper. Cambridge, MA: Massachusetts Institute of Technology - Sloan School of Management.

MacLean, Allan, Kathleen Carter, Lennart Lövstrand, and Thomas Moran. 1990. "User-tailorable Systems: Pressing the Issues with Buttons." in *Human Factors in Computing Systems*Seattle, Washington: ACM-SIGCHI.

Malone, Thomas W., and Kevin Crowston. 1994. "The Interdiciplinary Study of Coordination." *ACM Computing Surveys* 26 (1):.

Malone, Thomas W., Kevin Crowston, Jintae Lee, Brian Pentland, Chrysanthos Dellarocas, George Wyner, John Quimby, Charley Osborne, and Abraham Bernstein. 1997. "Tools for inventing organizations: Toward a handbook of organizational processes." .CCS-Working Paper. Cambridge, MA: Massachusetts Institute of Technology - Center for Coordination Science.

Nonaka, Ikujiro, and Hirotaka Takeuchi. 1995. *The Knowledge Creating Comapny: How Japanese Companies Create the Dynamics of Innovation*. New York: Oxford University Press.

Van Alstyne, Marshall. 1997. "The State of Network Organization: A Survey in Three Frameworks." *Journal of Organizational Computing* 7 (3):.

von Hippel, Eric. 1996. "Do It Yourself versus Specialization: Customization of Products and Services by Users of ASICs and CTI." *Management Science (forthcoming)* .