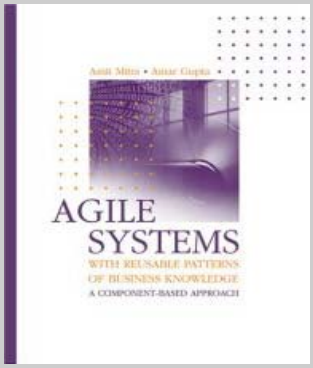
# *Agile Systems* by Amit Mitra and Amar Gupta

**Paul Harmon**

January 2006




**Agile Systems With Reusable Patterns of Business Knowledge: A Component-Based Approach**

**Amit Mitra and Amar Gupta**

**Artech House, 2005. $95.00, 382 pages**

Analyzing complex business processes is hard work. It's time consuming to determine just how an existing process works, to identify what people do, to determine the flow of information that guides the process and to define the business rules that are used to make decisions as the process is executed. Most business analysts who have been involved in a large business process analysis effort have thought that it would be nice if the effort could be made more efficient. Most of the analysts who have actually worked on simplifying business process analysis have focused on reuse – of identifying some process element that can be reused from one process to the next, or at least reused when similar processes are encountered.

**Business Process Frameworks**, like the Supply Chain Council's SCOR, base reuse on a process map that describes generic, high-level processes. Thus, SCOR practitioners believe that every supply chain can be conceptualized in terms of Source Processes, Make Processes, Deliver Processes and Return Processes. Once you realize you have a Source Process, then SCOR suggest what subprocesses you must have, how those subprocesses can be measured and what best practices are for those subprocesses. Frameworks speed the modeling and initial measurement effort.

**Software Components and Modules**, like the ERP modules from SAP or the CRM modules from Siebel, offer reusable blocks of automation. By picking and choosing, a process team can design an automated process that can be assembled from pre-existing software components.

A third option, described in Agile Systems, proposes the reuse of an established ontology, coupled with established business rules, and implemented in an software-based inferencing system. This is an option initially explored in the Eighties by those who, at the time, were concerned with the development of expert or knowledge-based systems. In essence, the first expert systems used rules to capture the knowledge of business experts and then made that knowledge available to other experts by putting the rules into a software system that, given information about a specific problem, could make an expert-level recommendations. As the early expert systems got larger, it was determined that rules alone were too clumsy. Hence, by the mid-Eighties, most of the more sophisticated expert system-building tools incorporated objects (they were called Frames in those days).

In essence, the objects in a sophisticated expert system-building tool formed a network that described the vocabulary of problem, and rules were added to reason about the facts as they were accumulated by the system. When one used these more sophisticated expert system-building tools, one began by accumulating knowledge from experts. Thus, if one wanted to build an expert systems to assist with home loans one would begin by working out the vocabulary of loans. One would probably identify vocabulary objects like: Home, Payment, Credit, Interest, Calendar, etc. Payments would probably have attributes like down payment and monthly payment, while Credit might have attributes like income, credit history, etc. In other words, one would construct a cognitive model of all of the concepts or words that a load officer typically used. Questions,

in effect, that the load officer would ask. Then one would begin to add rules that could reason with the information one had about a specific case. If the individual's credit history was superior, and her salary was $130,000 a year, and she could make a down payment of $50,000, what type of loan would she qualify for?

In other words, the objects and rules formed an abstract model of the concepts and rules an expert would use to organized knowledge about a particular subject and to reason about it to reach conclusions. The more complex knowledge-based systems depended on hierarchies that defined very high-level concepts that were then inherited by mid-level concepts. Thus, for example, a systems might have concepts like Thing and Information at the top level, and then subdivide Thing into subconcepts like Material Thing and Person. Person, in turn, might be divided into Customer, Partner, and Employee. It might also be divided into Male and Female. Thus, these concept hierarchies could get quite complex. The goal, however, was to establish fundamental relationships that would apply to all conceptual systems and to capture the rules and constraints at these abstract levels so that lower level concepts could simply inherit the appropriate attributes. Thus, for example, when I see a Robin, I don't have to study it in too much detail. I realize its a bird, and a living thing, an know a huge amount about it simply because I know it inherits all of the characteristics of birds and living things.

Amit Mitra and Amar Gupta propose to apply what I think of as an expert or knowledge-based systems approach to business process modeling. In essence, they would build a very abstract object model that would encompass all business concepts, and then build more specific networks defining the vocabulary and business rules of major areas of business – say buying and selling. If one then sought to create a business process in the area of sales, one would, in essence, create process objects that would inherit information from the more generic buying and selling model, which, in turn, would inherit from the general network on business when they term the Universal Pattern. Mitra and Gupta refer to their mid-level networks as reusable patterns of business knowledge.

In Agile Systems they begin by proposing a Universal Pattern that includes objects like Event, Fund, Energy, Physical Object, Person or Organization, Place and Information. They work out the basic attributes of these objects and define some of the rules or constraints that apply to them. Then they start to create mid-level neworks for more specialized business activities. They consider, for example, a shipment and transportation cluster, a document and information cluster, a task-resource cluster, a meeting and agreement cluster, and a buying and selling cluster.

Later, Mitra and Gupta propose, for the future, a knowledge machine. In essence, it would be a huge expert system that had all the knowledge of all the terms used by businesses and all the critical constraints or business rules. Anyone with a specific process problem would define the process, determine what elements of the process inherited what vocabulary, and instantly get an analysis of all the considerations and rules that might apply.

Along the way to the knowledge machine, Mitra and Gupta have explored all the technical problems one faces in creating this type of inheritance hierarchy. This kind of system can rely on the simple inheritance one finds in simpler object-oriented languages. It requires that one object can inherit from multiple parents, and that some objects can inherit some features but not others from a given parent. These are programming problems the bedeviled the expert systems designers in the mid-Eighties and they still create technical and conceptual problems today. I mention this only to suggest that this book is not light reading. It not only offers an survey of the high-level vocabulary and concepts of business, but a survey of some very complex programming concepts as well. (A second book is planned to offer even more technical detail.)

It's hard to imagine that anyone, in the near future, is going to build a real Knowledge Machine that could do what Mitra and Gupta propose. If anything, one can imagine a group working on such a machine for a specific domain like finance, or transportation systems. Thinking about what would be required to build a generic Knowledge Machine, however, is interesting. Just trying to identify the high-level business clusters or patterns is quite a challenge and very thought provoking.

In many ways Mitra and Gupta are discussing issues that are also being discussed in the technical committees of the Object Management Group (OMG), and elsewhere. There is a lot of interest in how one defines a comprehensive business ontology, how one organizes complex sets of business concepts and business rules, and how one uses meta-knowledge to inform specific process analysis projects.

This isn't an easy book and only those who are inclined toward ontology systems and logic will be able to make their way through this book. If you are such a person, however, particularly if you have a background in knowledge-based systems, you will probably find this book both interesting and insightful. Mitra and Gupta have thought long and hard about how to extend knowledge-based concepts into the general business arena, and I'm confident that we will be seeking more of this kind of thinking in the future.