

▶ **Business Modeling with UML: The Light at the End of the Tunnel**

by [Bryon Baker](#)

Product Manager
Requirements Management
Curriculum
Rational University

In the current economic climate, no software development project can afford to solve the wrong problem. Especially in companies for which a software system is the business, the cost of failure could be the solvency of the business itself.

How to ensure that you're solving the right problem? In its influential CHAOS Report, the Standish Group¹ specifies a firm, basic requirements definition and a clear statement of business objectives as essentials for project success. Without a business model as a focal point for insight and discussion around business objectives, it is difficult to develop an essential understanding of exactly what processes a software system must support. Just ask five different people in your organization how the business operates; you'll get five different answers. Hence the value of a business model: It provides consensus on the problem you're trying to solve.

So, great, you're committed to modeling your business before you undertake that next big technology project. But why choose the Unified Modeling Language (UML) as the basis for your business model? We'll explore its virtues for this purpose in this article.

What Is Business Modeling?

In a generic sense, business modeling is a set of activities whose goal is to help you visualize and understand business processes. As applied to software systems or other systems, business models act as a blueprint that will guide you as you construct the system. In effect, the model



- ▶ [subscribe](#)
- ▶ [contact us](#)
- ▶ [submit an article](#)
- ▶ [rational.com](#)
- ▶ [issue contents](#)
- ▶ [archives](#)
- ▶ [mission statement](#)
- ▶ [editorial staff](#)

becomes an operational description of the business that can illuminate value/cost tradeoffs, priorities, and risks. This level of understanding is frequently essential to helping system analysts, designers, and developers make informed decisions about the processes they're automating and the technologies most appropriate for implementing them.

There are three basic reasons why you might need to model a business:

- **To re-engineer a business.** This involves analyzing and fundamentally re-thinking how the business operates and interacts with the outside world. For this highest-risk form of process and system design, business modeling is absolutely essential.
- **To improve a business process.** I think of this as "bite-sized" process re-engineering, in which you identify an area of the business with the most critical problems and target your analysis to that area. The goal here is usually to streamline how the business works, and/or to enhance its competitiveness. Its micro-focus means it usually entails lower risk and a higher success rate than re-engineering.
- **To automate a business process.** This is the level of endeavor we customarily associate with software development.² The goal here is to reduce the resource requirements associated with a process by enabling more of it to happen without human intervention. In this context, a model of your current business allows you to understand the environment in which a software system will function.

Whether you plan to re-engineer the business or just automate an existing process, business modeling is the first step toward defining a software system that will solve the precise business problem you want to address. If yours is an established small company with a simple organizational structure, or if your team is attempting to automate or improve a very well-understood business problem, then you may realize few benefits from business modeling. But say yours is a startup operation, or you're entering a new business area. Here, modeling can provide critical insight into where automation can deliver the greatest ROI. If you plan to work with legacy systems or manual processes, or if you work in a large enterprise that is attempting to automate a mission critical process, then business modeling is almost certainly among the best things you can do to support project success.

Who performs business modeling? If the goal is to re-engineer the business, then modeling is most often undertaken by business process analysts who will develop new business architectures. If the goal is process improvement, then modelers may be business designers who need to describe new business processes. For a typical software development project, responsibility for modeling may fall to requirements analysts, whose job is to validate system requirements

The Business Value of UML

UML is generally thought of as a language that helps you visualize,

specify, construct, and document software-intensive systems. However, the designers of the UML made it possible to enhance the language so that it can be applied to many different domains. This feature has enabled it to rapidly emerge as the premier language for both traditional business modeling and the system analysis and design that follow.

UML has been successfully applied to the modeling of just about any system you can think of, from data structures to embedded real-time systems, to XML (Extensible Markup Language) schemas, and to real-world organizations from family businesses to multinational enterprises. So it's not surprising that business analysts find UML very handy for visualizing organizational processes.

But UML is far more than just handy; it's the most powerful, flexible notation available for business modeling today. It helps you manage complexity, reduce development time, and improve system quality. And there are six main reasons why:

1. UML provides a common language for business analysts and developers.
2. UML is visual.
3. UML is object-oriented.
4. UML describes business processes both structurally and dynamically.
5. UML helps you focus on the customer.
6. UML helps you derive better system requirements.

Let's look at each of these reasons more closely.

1. UML Provides a Common Language for Business Analysts and Developers

UML enables you to model business processes using the same symbols, diagrams, and other forms of notation that software teams use to model the systems to create or automate those processes. This ability to work using a common language enables something that was not previously possible in software development: Business people and systems people can communicate!

The idea sounds simple enough, but until UML was applied to business modeling, there was always a disconnect between the design of the business and the design of the systems within it. The UML largely eliminates that separation in perspectives between developers, business management, and customers.³ When you start with a UML-based model, key business considerations are more likely to be included in the system requirements, and that ultimately leads to a system that serves customers better.

2. UML Is Visual

In my opinion, the perspectives offered by flowcharts and spreadsheets are too linear to effectively model a business. A limited viewpoint can yield false impressions of what drives a process, where it is bottlenecked, or how information flows. To fully model a business, you need to answer time-honored questions about who, how, what, why, and when things are happening.

UML allows you to visually model how your business operates. The *who*, the *what*, and the *how* are all represented in terms of symbols and diagrams. In this context, relationships, activities, and the flow of information, goods, and services all become more obvious. These visual representations enable you to see bottlenecks, understand how information flows (or doesn't), and determine who does what with your business information. For example, a visual model makes it plain when too much information must flow through a single point, pointing to a potential need to redirect some of that flow to other processes.

A well-constructed visual model of a business -- such as those the UML makes possible -- will answer all of these fundamental questions for you:

- Who are your internal and external customers? (I.e., Who will benefit from this business system?)
- Where can a system best add value to your business?
- What events within and/or outside the organization trigger each business process?

OVERVIEW OF BUSINESS MODELING WITH UML

Let's take a quick look at how you construct a business model with UML, with an eye toward how the process delivers business benefits.⁴ Though tremendously powerful, a UML-based business model is conceptually straightforward. It consists of two key elements:

- A business use-case model, which describes the actions (i.e., a workflow) that a particular business process performs in order to deliver value to a business customer.
- A business object model, which describes how a business process will accomplish the actions described in the business use-case model. This model helps you see how people and things (goods, information, etc.) are related, and how they interact to perform the process in question.

Other key concepts include the *business worker* and the *business entity* (Figure 1). A business worker represents a role or set of roles in the business. A business worker interacts with other business workers and manipulates business entities, while participating in business use-case realizations. A business entity represents a "thing" handled or used by the business.

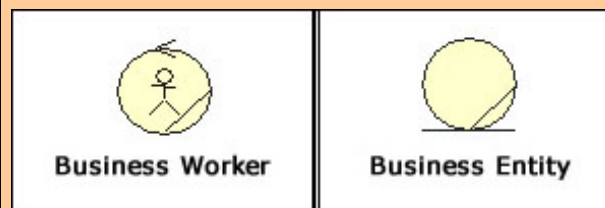


Figure 1: Business Worker and Business Entity in UML Notation

In general, a business use-case describes what the business does (i.e.,

- What end products do those business processes produce?
- What internal deliverables do those business processes entail?
- What is the organizational structure that supports the business?
- What roles and responsibilities within that organizational structure should be part of the system?

Having answers to these questions -- and the ability to follow up on related concerns -- will help you gain valuable insight into whether the changes you're about to make will solve the right problem, and do it in the way that delivers greatest value to the customer.

3. UML is Object-Oriented

UML is used to diagram software systems from an object-oriented perspective. When you model a business with UML, your business concerns are clearly outlined relative to the appropriate *business objects*, in the form of a *business object model* (see Sidebar).

Objects are entities that parallel objects in the "real world": They have some properties (such as name and address), relate in various ways to other things around them, and will exhibit some sort of behavior when acted upon. Object-oriented models can therefore very closely approximate actual business objects and systems, even to the extent of portraying how different parts of the system work together dynamically. This is true, in a nutshell, because

what it delivers to its customers); a business object model describes how it does it. You specify business use cases first, and then use these to derive the business object model. A business use-case specification takes the form of a text description, along with one or more UML diagrams. A business object model can include class diagrams, activity diagrams, and business interaction diagrams -- all of which depict relationships and interactions among the entities that perform the activities of the business.

Figures 2 and 3 illustrate business use-case and business object models as applied to a business process automation problem. Note that they include two types of actors: Business actors (black stick figures with a yellow face crossed by a line) show interaction with business use cases; system actors (red stick figures) show interaction with system use cases.

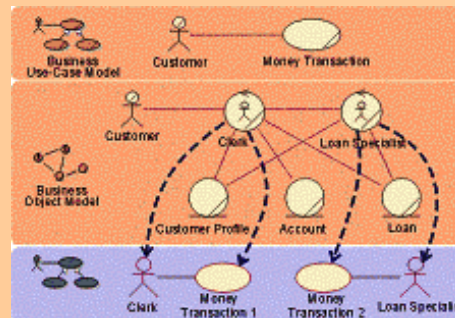


Figure 2: Business Models and Actors to the System

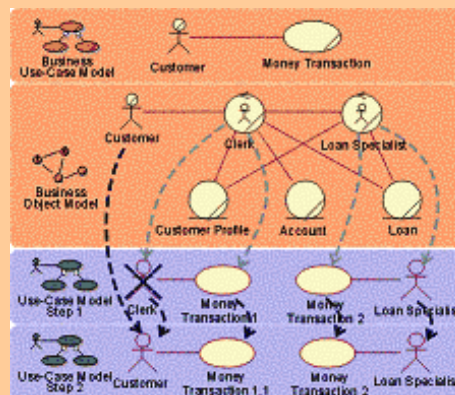


Figure 3: Automating a Business Worker

Here you can clearly see how the work you invest in producing a business

the objects that comprise UML models reflect *real-world* entities very closely. This means object-oriented models are therefore more tangible and well described, and overall more flexible and intuitive.

model will help you develop your system requirements.

Take, for example, an object-oriented description of a worker's role. It would likely include information about behavior, such as job responsibilities with respect to other workers in the business; state, such as how the role relates to other objects and to the process; and properties, such as quality checklist items or a job description.

4. UML Describes Business Processes Both Structurally and Dynamically

The more automated a business becomes, the more its interrelated software systems form the core of the business value it delivers to customers. Understanding how those interwoven systems interact -- and how to guide their successful evolution in response to a changing business landscape -- is critical to success. The value of UML in this context is that it lets you look at things inside the business both structurally and dynamically. The UML has many different diagram types that enable you to represent information from many different angles. These orthogonal views will completely describe your business and provide *meaningful* information to the varying levels of stakeholders.

UML use cases (see Sidebar) describe business processes from the viewpoint of how actors (*customers* for our purposes) use the business to attain some goal. A use case will describe a complete business process from start to finish, from an external perspective. This is in direct contrast to many "traditional" business-modeling methodologies, which *decompose*, or break down, processes along functional lines.⁵ In such a decomposition, the whole is not necessarily the sum of its parts. The descriptions of the processes become isolated from each other, and their relationships become less apparent. When this happens, the benefit these isolated descriptions have to the business are less quantifiable, and their value therefore starts to become subjective.

The value of UML becomes especially evident when you actually turn a business modeling process loose on a business problem. With conventional methodologies, key issues can be easily misinterpreted, because these methodologies often overlook dynamic relationships that impact the process.

UML, in contrast, provides rich alternatives for describing the dynamic, real-world connections that are part and parcel of modern business systems. The multiple views that a UML model provides really help you understand a problem from all sides. Also, because UML provides a common language and a shared basis for discussion among development and business staff, there's no need to translate the model into words and risk diluting its meaning.

5. UML Helps You Focus on the Customer

The UML business modeling methodology revolves around business use cases, which emphasize how a business process delivers value to the customer. This customer-centric perspective helps you conceive an external view of the system you're creating. This is an especially powerful and useful approach in e-business contexts, where an external focus is even more critical to success.

In UML, a business use case is defined as:

A sequence of actions performed by the business that yields an observable result of value to a particular business customer.

This parallels the influential work of Hammer and Champy (1993),⁶ who define a business process as:

A collection of activities that takes in one or more kinds of input and creates an output that is of value to a customer.

Business processes are what drive automated business systems; they comprise the tangible behavior of the organization. Because business processes are the means for delivering value to customers, the success of an organization hinges on the performance of its business processes (whether they are automated or not).

UML business use cases are built from the ground up to tell a story about how a business process works and how a customer uses that process. What better basis upon which to design a software system for optimal process performance -- and ultimate business success?

6. UML Helps You Derive Better System Requirements

With its rich descriptions of relationships among components, business actors, and other entities, UML makes it far easier than other modeling approaches to identify where a system best fits within a business context. This, in turn, enables you to more readily validate your software requirements. The end result is a working business system that solves a specific business problem, meets the real needs of the business, and delivers optimal value to customers.

Moreover, the UML-based business models your team creates can serve as direct input to a requirements model. The Rational Unified Process® (RUP®) provides a direct mapping between a UML business model and a requirements analysis model. This level of integration "front loads" your system analysis and design efforts, and further compresses time to deployment.

The Bottom Line

The nature of software development is changing in response to new business demands. Everywhere you look, innovative organizations are re-thinking traditional business processes and extending key internal processes beyond organizational boundaries to reach suppliers, customers,

partners, and government agencies.

In a world in which the risks associated with the failure of key processes has never been greater, it's no wonder business modeling is catching on like never before. When business requirements are poorly defined, the resulting inadequacies compound themselves and reverberate throughout the integrated enterprise. With a clearly explicated model of the business, however, companies have a foundation upon which to build and deliver successful systems -- that ultimately deliver optimal value to customers.

UML surpasses all other alternatives for developing a solid business model that can be applied to the development of a successful software system:

- It unifies development and business teams by integrating the modeling and development languages across every phase of the effort.
- It facilitates the specification of a robust, component-based architecture whose design and development can be controlled, managed, and verified, leading to reduced costs and shorter development cycles.
- It helps, at every step in the process, to keep you focused not just on technical innovation -- or even greater efficiency -- but also on the value you're delivering to customers.

Notes

¹ www.standishgroup.com

² Business process automation is also the focus of the business modeling discipline in the Rational Unified Process® (RUP®).

³ A *customer* is someone or something that interacts with the business, via a business process, for the purpose of obtaining something of value from that business. This can be an end customer, a supplier, a trading partner, a prospect, a federal regulatory agency, or another part of the business that is internal to the organization. It all depends on where you draw the boundaries of your business modeling effort.

⁴ For a more complete introduction, see "[Introduction to Business Modeling Using the Unified Modeling Language \(UML\)](#)" by Jim Heumann in the March, 2001 issue of *The Rational Edge*.

⁵ It is also worth noting here that business modeling using the UML and use cases helps support an iterative development process. Whereas functional decomposition techniques are not, because they require you to functionally decompose the entire business area of interest before you can move on.

⁶ M. Hammer and J. Champy, *Reengineering the Corporation*. HarperBusiness, 1993.

References

Ivar Jacobson, Maria Ericsson, and Agenta Jacobson, *The Object Advantage: Business*

Process Reengineering With Object Technology. Addison-Wesley Object Technology Series, 1995.

M. Hammer and J. Champy, *Reengineering the Corporation*. HarperBusiness, 1993.

Magness Penker and Hans-Erik Eriksson, *Business Modeling with UML: Business Patterns at Work*. John Wiley & Sons, 2000.

Rational Unified Process, Version 2002.05.00.



For more information on the products or services discussed in this article, please click [here](#) and follow the instructions provided. Thank you!

Copyright [Rational Software](#) 2001 | [Privacy/Legal Information](#)