SOFTWARE DEVELOPMENT MAGAZINE: MANAGEMENT FORUM

December, 1999. Vol. 7, No. 12

Capturing Business Rules

By Ellen Gottesdiener,

[Editor's Intro]

With our noses to the software development grindstone, it can be hard for us to see much of the business world around us. One way or another, the people for whom we and our developers create software are in some kind of business, a business with business rules. Just as the languages and operating systems within which we work function by a complex collection of implicit and explicit rules, so, too, do the business operations of our clients and customers. We, in turn, have our own business rules. We may expect to be told what we are building before we are ready to supply an estimate of how long it will take to build it, and we expect our customers to understand such rules of the game. Not surprisingly, they expect something similar from us. To support them with good software, we need to understand the rules of logic behind the businesses we are serving. This month, guest columnist Ellen Gottesdiener makes the case for business rules as requirements.

—Larry Constantine

Business rules provide the knowledge behind any and every business structure or process. They are therefore at the core of functional requirements.

An inept inadequate or inefficient requirements analysis starts a project on the wrong foot. Time is wasted at a point when time is of the essence, and developers will find it hard to produce a good software product that meets customer needs. Capturing, validating and verifying functional requirements are major challenges — not only for clients and business partners, but also for managers and software developers. Developers need clear and usable requirements that can help guide the product through the development process .

The requirements process includes whatever techniques are used to capture them (interviews, facilitated workshops, prototyping, focus groups or the like; what tools are used for capturing and tracing information (text, diagrams, narratives or formal models); and how customers and users are actively involved throughout the process. When it comes to the requirements product, management concerns include the testability of functional requirements, the ability to find and resolve conflicts in requirements, the ability to link requirements back to business goals (backwards traceability) and the ability to track functional requirements throughout the life cycle from design to code to test and deployment (forward traceability).

As experienced managers, we do not expect perfect solutions for requirements engineering. However, in my experience there is at least one "secret weapon." Whatever functional models you wish to use, whether use cases, CRC cards or some proprietary technique championed by your boss, the important thing is to focus on the true essence of the functional requirements: the business rules.

Rules rule

Business rules are the policies and constraints of the business, whether the “business” is banking, software development or automation engineering. The Object Management Group simply defines them as “declarations of policy or conditions that must be satisfied.” They are usually expressed in ordinary language and are “owned” by the business. Your business may be commercial, not-for-profit or part of a government, but it still has business rules. They provide the knowledge behind any and every business structure or process. Business rules are also, therefore, at the core of functional requirements. You may use various functional requirements models—structural (data, class); control-oriented (events, state-charts); object-oriented (classes/objects, object interactions); or process-oriented (functional decomposition, process threads, use cases or the like). But within the heart of all functional requirements are business rules.

Business rules are what a functional requirement "knows": the decisions, guidelines and controls that are behind that functionality. For example, functional requirements or models that include processes such as “determine product” or “offer discount” imply performance of actions, but they also embody knowledge—the underlying business rules—needed to perform those actions. An insurance claim in a “pending” stage means that certain things (variables, links to other things) must be true (invariants, or business rules). The behavior of a claim in a pending life-cycle stage is, thus, dependent upon business rules which govern and guide behaviors.

As the essential ingredient of functional requirements, business rules deserve direct, explicit attention. Since business rules lurk behind functional requirements, they are easily missed and may not be captured explicitly. Without explicit guidance, software developers will simply make whatever assumptions are needed to write the code, thus building their assumptions about conditions, policies and constraints into the software with little regard for business consequences.

When the rules are not explicit, and if developers encode them by guessing, the essential business rules may be discovered as missing or wrong during latter phases.. This results in defects in those later phases that could have been avoided if the rules had been baselined during requirements analysis. In the end, the lack of explicit focus on capturing the business rules creates rework and other inefficiencies.

Rather than just mentioning business rules as “notes” in your models, you should capture and trace them as requirements in and of themselves. Focusing on business rules as the core functional requirements not only promotes validation and verification, but it can speed the requirements analysis process.

Top-down, again

In working on a business-rule approach to software development, I have come to realize that, ideally, such an approach needs to be driven from the top down, like the traditional top-down methods of information engineering. Unfortunately, in the modern world, with business moving at the speed of a mouse click, a systematic, top-down analysis is often an unaffordable luxury. Consequently, I’ve become more practical; a business-rules approach doesn’t need to be a method or methodology in and of itself. Rather it is just a necessary part of requirements engineering. It includes a process (with phases, stages, tasks, roles, techniques, etc.), a business-rule meta-model, and business-rule templates.

Advocates of the recently standardized Unified Modeling Language (UML) and the accompanying all-purpose "Unified Process" toss around the term “business rules” in presentations and conversations, but neither UML nor the “unified process” offers much guidance for business rules. UML has an elaborate meta-model and meta-meta-model to support its language. One of the classes at the meta-model level is called “Rules.” But the UML has given business rules short shrift. The only modeling element that is practically usable is the “constraint” element, which can be attached to any modeling element. Furthermore, although the UML’s OCL (Object Constraint Language) is a fine specification-level language to attach business rules to structural models, it is not a language to use during requirements analysis when working directly with business customers to elicit and validate business rules. Business rules need to be treated as first-class citizens.

Use cases and business rules

At the center of the UML are use cases, which are viewed by some analysts as business requirement models, while others call them “user-requirement models.” Use cases are a functional (process) model that can be expressed as words using a natural language; some templates also may include special sections, such as pre- and post-conditions, goal name and the like. Use cases have proven to be an important and useful model for capturing requirements. Recent work has evolved use cases from an often vague requirement deliverable into something specific, focused and usable. Alistar Cockburn has directed attention to the goals of use cases, and Larry Constantine and Lucy Lockwood have developed essential use cases for usage-centered design. However, use case narratives—the flow of events, as UML people would say—are all too often missing the very essence of the use case, because behind every use case are business rules at work.

Failing to capture and verify the business rules which underlie the use-case models can lead to the delivery of a software product which fails to meet the business needs. Formalizing business-rule capture concurrently with use-case model development strengthens the delivered product through the synergy between use-case models and business rules.

Business rules come in many forms. I think of them as terms, facts, constraints, factor clauses and action clauses, but there is no agreement on a standard taxonomy or set of categories for business rules nor should there be. The taxonomy should fit the problem. Some things in the “problem space” are more business-rule-based (e.g. underwriting, claim adjudication, financial-risk analysis, medical instruments monitoring). Other problems are more business-rule-constrained (payroll, expense tracking, ordering, etc.). This requires the selection or tailoring of a taxonomy and an accompanying business-rule template for any given business problem. The template provides a standard syntax for writing business rules in natural language. Such tailoring is beneficial since it requires us to understand the problem in greater depth by working directly with our business customers to perform this tailoring and to derive an appropriate business rule template.

Business rules can be linked to a use cases and to requirements statements. Business rules also can be linked to other models, depending on the depth of requirements traceability and the importance of specific attributes, such as source, verification process, documents, owner or risk. These other models can include glossaries, life-cycle models, activity models and class models. The business rules are thus reusable across multiple types of functional requirements and can be traced along with other requirements.

Besides use cases, other models can be useful for identifying and capturing business rules. For example, lifecycle models, such as a simple state-chart diagram, can be quite usable and understandable to business analysts, especially for problems in which a core business domain has many complex states. Even object-class or data models can be excellent starting points for problems that require understanding the domain structure, but which do not require a lot of "action" to take place.

Collaborate with customers

Business-rule discovery and validation requires knowledge of the thinking processes of the business experts from whom we elicit functional requirements. Collaborative work in groups that include customers is most effective in the early lifecycle phases of planning and requirements analysis, as well as for ongoing project process improvement. Such collaborative work patterns can be used effectively to model business rules and derive higher quality requirements, which include business-rule requirements.

In the collaborative approach to business rules that I prefer, requirements analysts and business customers collaborate to create a business-rule template that is expressed in natural language, based on common sense and which is directly relevant to the business customer. In modeling use cases and user interfaces, business rules are explicitly derived and captured using this natural language template. After all, what makes sense to our customers and users is what we express in their own language.

Collaborative modeling of business rules can be a powerful, eye-opening process. It is amazing to see customers realize that their own business rules are unclear, even to them. Often they come to realize the rules are not standardized and therefore may be inefficient or even risky. Considering regulatory exposure and potential for lawsuits, collaborative modeling can convincingly make the case for immediate clarification of business rules.

Eliciting business rules can be a real challenge, especially when business experts do not agree on the business rules or when the rules are unknown or very complex. I find that facilitated requirement workshops (see accompanying story in this issue) in which business rules are explicitly tested within the context of modeling other things—such as use cases—is the most direct and productive tool for eliciting and validating the rules of the business. The workshop process exploits the power of diverse people joined together for a common goal. In workshops or even in interviews with business experts, if the problem domain is very much 'rules based' (vs. rule constrained as mentioned earlier), using cognitive patterns is extremely helpful to accelerate the group's ability to express the rules. Such patterns, with their roots in knowledge engineering, model business expert's thinking process and enable the rules to emerge.

Some customers may wish to take on a business-rules approach because they want to uncover the real "dirt" of the business. They are ready to ask the "why" of the business rules. Actually, the question of larger purpose should be asked of any functional requirement. If the answer does not map to a business goal, objective or tactic, then the functional requirement is extraneous. Business rules exist only to support the goals of the business. Whereas good use cases represent the goals of external users, the business rules behind use cases are inextricably linked to the goals of the business itself. If not, the rules are extraneous and may even be in conflict with business goals. Thus, mapping business rules to business goals is a key step in validation and promotes traceability back from business-rule requirements to the business goals.

The business-rule cure

Business rules are an ounce of prevention. Unless we get them, get them right and get them early, we are destined for problems in our projects and products. The project problems stemming from incomplete, erroneous or missing business rules include redefining requirements and re-testing results. The product problems are worse because, if the rules are wrong, in conflict or redundant, the users of the software product suffer. Unless we get to the very heart of the business during requirements analysis, with the business rule written in text by business people themselves, we are doomed to passing incomplete, inconsistent and conflicting business-goal requirements ahead to production. Thus, we must get to the very essence of requirements with business rules, written by business people.

To get to the heart of the matter, active business sponsorship is absolutely required. The process can be acutely uncomfortable because business-rule capture and validation exposes the "undiscussables"—the unclear and conflicting business policies and rules. It also begs for rethinking the sub-optimal rules and requires the realignment of the business rules with the business goals. To resolve such issues requires business sponsorship and leadership. To go forward with a business-rules approach in the absence of such sponsorship is treading on very thin ice. Only the collaborative efforts of IT professionals and their business partners with the active involvement of business management can yield the benefits of a business-rules approach in quickly and succinctly cutting to the core of the functional requirements.