Rule Engines

For more complex calculations with many variables we find the so-called rule-based systems as the most suitable solution. Instead of entering an explicit algorithm for a calculation which can change dynamically in time, a set of rules is defined to calculate the result based on input parameters. The rule-based systems are well suited to calculate tariffs, fees, identify fraud, score customer behavior or determine the amount of reductions, bonuses or commissions offered, or various combinations thereof, and they can also take into account different groups of customers or users, as well as other constraints. The administration of such rules is simple and intuitive for operators with the relevant business knowledge, and requires neither the knowledge of the algorithms used, nor any programming skills. Recently, we also started using rule-based systems to manage complex user-interface workflows, with favourable results.

Rule Systems

Exceptions are the rule in all enterprise software. Especially custom-developed, company-specific parts tend to be full of IF-THEN statements, often combined into elaborate constructs. Such software - often called "business logic" - is intimately related to the circumstances and internal workings of the business, perhaps specifying procedures that would seem totally arbitrary to an outsider. At the same time, it counts as critical infrastructure of today's enterprise: errors in it may cause production downtime, losses of customer goodwill and revenue, or even legal liability. Finally, business logic is often scattered across many layers, components or even distinct software systems. The upshot: changes to business logic tend to be difficult, risky, expensive and slow. The situation may get worse over time as knowledge embedded in the software becomes diluted (people move on, new ones do not have enough context etc.). Business logic software may eventually constrain the company's agility in the marketplace as necessary adaptations become too expensive to be relevant. Rule systems in their current manifestations represent remarkable progress towards solving these vexing problems.

The two fundamental concepts in rule-based reasoning are facts and rules. Facts are individual pieces of information relevant to the business domain. These may range from the very simple to quite complex: a fact may consist of an order line-item or an insurance contract or even a complete patient record gathered over several years. Rules are essentially familiar IF-THEN sentences defining actions that the IT system should take when certain facts become known. Each rule consists of a filter identifying which facts or combinations of facts the rule should react to ("the left-hand side"), and commands specifying what should happen ("the right-hand side"). A really mundane rule, for instance, may say "IF a cart contains items totaling at least 100,00 € THEN mark it up for a 5% discount".

What distinguishes rules from ordinary computer programs is the way they are processed. Rules are interpreted by a special program called rule engine. The rule engine keeps the set of all rules, called rule base, in a context called working memory. When a client (an outside program) inserts a fact into the working memory, the rule engine tries to match it against the left-hand sides of all rules. Each time it finds a matching rule, it performs the actions specified in its right-hand side. That may affect the working memory - adding, changing or removing facts - which triggers another round of matching. A fact may thus launch a cascade that would be quite difficult to describe in a conventional program but happens very naturally in the rule engine (imagine a pile of pebbles reconfiguring itself when a new pebble is added).

Rule-based reasoning allows for specifying fairly complex behaviors using conceptually simple building blocks. Rules are also generally quite easy to extract from process documentation or interviews with business domain experts; they correspond more directly to the fluid know-how in an organization than sequential descriptions of algorithms. They are consequently easier to comprehend for people without an IT background. Perhaps more importantly, a repository of rules potentially isolates and encapsulates business logic into a well-defined base of knowledge (this depends on how well the rule engine integrates with its client environment).

Of course, writing a rule base involves its own pitfalls and requires a certain degree of expertise. Being an expert in the core business domain of a company is definitely not sufficient for writing rules proficiently. Curiously, neither is programming expertise. People who are used to what's called "imperative programming" have to adopt a different way of structuring problems which may involve un-learning some very strong habits. Even basic questions such as estimating the correct granularity of facts or deciding which processes should be implemented as rule sets vs. left as procedural programs are non-trivial questions that are best tackled with some amount of hands-on experience or the assistance of a business-rule analyst. Another difficulty is that implementing a rule system usually uncovers lots of inconsistencies even in the thinking of domain experts, exposing additional rules, exceptions and special cases. That's actually beneficial for the organization in the long term but it does complicate the introduction of rule systems into the IT infrastructure.

Rule systems, like any tool, are useful in some contexts and less useful in others. When deployed properly, however, they have a powerful capability to transform the IT angle of important problem areas in the enterprise, turning them from liabilities into assets.

Plaut Slovensko, s.r.o
Karloveská 34, 841 04 Bratislava
Slovenská Republika
Tel.: + 421 2 3215 1411
Fax: + 421 2 3215 1412
E-mail: info@plaut.sk
Web: www.plaut.sk