Specification of Business Value with and in Software Patterns

Vladimir Tosic\textsuperscript{1,2,3}, Basem Suleiman\textsuperscript{1,2}, Abdul Babar\textsuperscript{1,2}
\textsuperscript{1} NICTA \& \textsuperscript{2} University of New South Wales, Australia; \textsuperscript{3} University of Western Ontario, Canada
vladat@computer.org, basem_suleiman@yahoo.com, abdul.babar@nicta.com.au

Abstract

Business value is a crucial, but frequently neglected, aspect of software quality. Specification of business value with and in software patterns promises improved software quality. Here, ‘with’ refers to patterns describing best practices in modeling, monitoring, and control of business value of software. Contrary, ‘in’ refers to annotations of existing patterns (for various problems) with business value information. We illustrate our discussion on the WS-Policy4MASC policy language and the UML profiles for it. WS-Policy4MASC enables specification of diverse business values and various business strategies for maximizing business values in run-time system monitoring and control. The UML profiles for WS-Policy4MASC can be used for specification of business value (and other quality aspects) both with and in software patterns.

1. Modeling of business value of software

Business value is a broad concept that refers to any measure of worth of a business entity [1]. It includes not only financial aspects (e.g., income, costs, profit), but also many other aspects (e.g., market share, customer satisfaction) important for business operations. Since the purpose of software is to provide benefits to its customers and users, business value is an extra-functional feature that is crucial for software quality.

Unfortunately, it is not easy to relate business value with technical extra-functional properties of software quality, such as performance, security, and maintainability [2, 3]. For example, higher system availability, lower response time, and/or more modular architecture need not lead to long-term increases in (vendor and/or customer) profits. Further, while there are many solutions for monitoring and control of performance or security, there are only a few for monitoring and control of customer satisfaction with software. While these problems have been known for decades and there were isolated attempts at their solution, only recently systematic approaches to modeling of business value of software and use of these models in software engineering processes [3], software engineering artifacts (e.g., designs) [3, 1], and solutions for run-time monitoring and control of software execution [2] started to appear.

Among such contributions is our WS-Policy4MASC, a new XML (Extensible Markup Language) format for formal specification of monitoring and control policies for Web service systems [1]. It extends the Web Services Policy Framework (WS-Policy, currently standardized) by defining new types of policy assertions (goal, action, utility, and meta-policy) and additional necessary information (e.g., events and schedules). Among its original contributions are specification of diverse business values (benefits or costs, tangible or intangible, agreed or possible, absolute or relative) and miscellaneous business strategies maximizing various combinations of business values (e.g., only agreed intangible benefits). These strategies can be used for run-time management (control) of software execution. In addition, WS-Policy4MASC enables specification of management information about functional constraints and quality of service requirements/guarantees and adaptation actions. Our language is used in the Manageable and Adaptable Service Compositions (MASC) middleware. To facilitate development of Web service systems that can be managed with WS-Policy4MASC and MASC and to improve alignment between run-time management activities and design-time models, we also developed UML (Unified Modeling Language) profiles for WS-Policy4MASC [1]. They improve support for: a) specification of diverse management information within design-time models, b) automatic creation of run-time management policies from design-time models, c) feedback of run-time management information values into analysis of design-time models.

2. Software patterns and business value

A software pattern describes an empirically proven abstract solution for a particular problem in a specific context and under certain constraints (forces), as well as when and how to apply this solution. Implementation of a pattern is, thus, a reuse of known best practices. This can lead to many benefits, including reduction of development time and effort and higher software quality and consistency. The use of patterns often (but not

\* NICTA is funded through the Australian Government’s Backing Australia’s Ability initiative, in part through the Australian Research Council.
always) increases business value of software. Relationships between business value and software patterns are complicated, but are important for software quality. We find that at least two broad, mutually orthogonal, categories of relationships can be discerned (they might be generalizable to the other aspects of software quality):

1) Specification of business value WITH software patterns: This refers to mining, formalizing, and using patterns describing best practices in modeling, monitoring, and control of business value of software. The main benefit is improved software engineering of systems that participate (as managing and/or managed entities) in business-driven IT management. For example, if a particular configuration of classes from our WS-Policy4MASC UML profiles is independently used at least 3 times to solve variations of the same problem, a software pattern can be defined as a reuse unit. Patterns for monitoring and control of different types of business value would also be useful. For example, monitoring of financial business values is now common (e.g., in resource consumption accounting/billing software) and there are a few published patterns related to this area. Recurring solutions to less common monitoring of non-financial business values (e.g., customer satisfaction) could also be described as patterns, but we are not aware of such publications.

2) Specification of business value IN software patterns: This refers to annotating (graphically and/or textually) existing patterns (for various problems) with additional business value information. The main benefit is to facilitate development of diverse software systems that leverage value-based software engineering solutions and can be more easily controlled by business-driven IT management. For example, UML descriptions of patterns can be annotated with stereotypes from the UML profile for WS-Policy4MASC. Similarly, various textual parts of a pattern description could contain a textual description of business values described in the WS-Policy4MASC model. These can be not only run-time business values of the described solution (e.g., cost of subscription to a mandatory external Web service), but also design-time business values of using this particular pattern (e.g., cost of patent licensing fees). Since WS-Policy4MASC also describes functional constraints, quality of service, and other management information, similar annotations can also be used for these aspects of software quality.

Formalization and storage of software patterns annotated with business value information is also beneficial for business value driven run-time adaptation. In WS-Policy4MASC, actions can be used to specify which pattern to use, while business value maximization strategies can be used to decide the most beneficial among several applicable patterns. When operational circumstances change during run-time and system configuration based on some pattern is no longer applicable, a pattern repository could be examined to find the most beneficial replacement pattern for the new circumstances. This new pattern would be instantiated and tailored (in the ideal, far-future, case: without human input) to adapt the running system. Since patterns are proven solutions, this reduces adaptation uncertainty.

Annotations describing quality aspects (including business values) can be in any section of a pattern description, but most likely in the Solution Description, Constraints (Forces), and Rationale. It might be beneficial to have a separate ‘Quality Aspects’ section in a pattern description, with a ‘Business Value’ subsection. This section would contain most of quality and business value information captured in a pattern, in order to make a stronger emphasis on quality aspects and to support pattern maintainability (quality aspects tend to change more frequently than other aspects). However, quality information logically cross-cuts different sections of a pattern specification, so it is likely that not all quality information can be put into this section.

While specification of business value with and in software patterns could lead to improved software quality, currently it does not exist (and similar works address only limited sub-issues). The main reason is that practical use of the existing works on modeling, monitoring, and control of business value (including MASC, WS-Policy4MASC, and our UML profiles) is still limited to small communities. For example, although we developed WS-Policy4MASC UML models of several systems and noticed recurring solutions, the requirement of 3 independent uses is not yet satisfied. For the discussed ideas to become a reality, the first necessary prerequisite is more widespread use of modeling of business value. Subsequently, systematic study of independent solutions to recurring problems and existing patterns in different areas will be needed.

References