Global Reuse Strategy Based on Use Cases

Michał Śmiałek
Warsaw University of Technology, Department of Electrical Engineering
pl. Politechniki 1, 00-661 Warsaw, Poland
InfoVIDE Ltd., ul. Kolejowa 5/7, 01-217 Warsaw, Poland
smialek@iem.pw.edu.pl

ABSTRACT
The paper introduces two concepts based on use cases: business patterns and automation patterns. It describes a connection between the business and automation patterns with classes. Finally, it presents an idea of a multi-level library that allows global reuse of software development models and code.

Keywords
Software reuse, use cases, development patterns, UML.

1. TRACEABILITY OF REUSE MODELS
One of the answers to the well known problems of incompatibility with the user's needs and long development lifecycles is the reuse of components that use standardized middleware technologies [4]. Yet there remains a question: having a library of reusable components, how do we find the components to build a system that satisfies the requirements? This paper tries to answer to the above question. It proposes an idea of software development strategy based namely on reusing the user's requirements. The strategy adds an additional "use case" level to the existing strategies (see e.g. [3]). The result is a multi-level library of reusable models denoted using UML [1].

Current reuse schemes offer the reuse of "artifacts" on two levels: component/class level, and analysis/design pattern level. A developer can browse through a library and seek for a component or pattern that could fit into his/her design. However, if we want to ensure compatibility with the user requirements, this browsing should be controlled by the requirements which in UML translates to use case driven approach [5]. In a use case driven reuse strategy we can of course reuse the use cases, their descriptions and use case diagram patterns. But we want also to reuse components or design patterns traced from these use cases. We therefore need a clear path (denoted in UML) leading from business use cases through system use cases to classes/components.

Figure 1: Trace from business use cases to automation patterns

Figure 2: Trace from system use cases to analysis/design patterns

The trace from business use cases lies in their business process description. The considered fragment of the business can be usually described with several scenarios that form an activity diagram (Fig. 1). Activities that make the activity diagrams seem to be the "missing link" between business use cases and system use cases. Mapping a single activity onto a single system use case gives us the first clear step on our traceability path. The second step is to map system use cases into classes that satisfy them. This is the standard "use case realization" with scenarios, sequence diagrams and activity diagrams (Fig. 2).

2. REUSE OF BUSINESS AND AUTOMATION PATTERNS
The reuse artifact on the business level can be called a "business pattern". The business pattern basically describes a fragment of a particular business domain. This description
The second level of the library allows us to choose a particular business use case. We can pick one found in a library business pattern. We can also try to match a business use case developed during our own business analysis. When we find one of the business use cases, we obtain a set of its automation patterns. Now we can choose one of the patterns most appropriate for our system.

On the third level we seek for system uses cases. Again, we can pick one found inside an automation pattern, or we can try to match a system use case developed during separate system analysis. Matching or picking a system use case, results in obtaining a set of analysis/design patterns appropriate for this use case. Sequence diagrams that show traces from the use case's scenarios to individual classes supplement the patterns.

The final fourth level of the library allows us to seek for components based on the classes/interfaces found on level above. On this level we deal with individual classes or interfaces. We can seek for a class specification that results in finding a particular code component that fulfills the specification.

As we can see on Fig. 4, the most important elements of the library are its search engines. The basic condition for effective use case matching are unambiguous scenarios. It seems that the best solution here is to use simple formal grammars as suggested in [2].

4. CONCLUSIONS

The proposed strategy can be a first step towards global reuse culture in the software industry. This culture could be based not only on the reuse of software components or even component patterns. It can be based on the reuse of business automation patterns that trace to specific solutions with ready components.

Globalization of the reuse strategy could mean that all companies describe their business with use case based business patterns. Automating such a business is then a matter of picking an automaton pattern for a particular business use case that we want to automate. Here we can see a field for competition between software companies - development of their own automation patterns with appropriate realizations.

5. REFERENCES


