Executive Summary

Refactoring in the Context of Enterprise Architecture

by Sebastian Konkol, Senior Consultant, Cutter Consortium

Although the necessity of refactoring is not being questioned by agile developers, its business justification has always been doubted by "outsiders." The accompanying *Executive Report* proposes a means of support for refactoring efforts and initiatives through enterprise architecture (EA), specifically, embedding the starting point of the refactoring process into the EA management process and other technology management processes.

OVERVIEW OF THE PROBLEM

The IT organization often struggles to balance the relatively small number of big projects aimed at rolling out a new IT system with the relatively big number of small "projects" aimed at the introduction of various small changes in the existing IT systems environment. With each scale of development effort — whether it is a big project or change request — some constraints and limitations are being introduced that are rooted in the necessity to compromise software flexibility or architectural purity in favor of constraints like time allowed for the work to be done or available funding. The smaller the development effort, the more likely it is that external constraints will result in the software limitations with respect to initial requirements.

In such an environment, there is no time (or, at best, a very limited amount of time as well as resources) for

software refactoring; the business seems unwilling to accept the fact that, once developed, a solution may have to be altered due to other changes implemented into the information systems (nor does the business want to pay for such changes). Items that initially were implemented as locally optimal and that should be corrected "the sooner the better" often are uncovered over and over again during the development work of a subsequent change implementation. Thus, we have a vicious circle.

The report begins by providing some rationale and a framework for the business justification of the proposed approach; it defines the concepts, refactoring-problem lifecycle, and governance rules and shows how to embed them in EA management processes. The report provides some ideas of where to start when implementing the concept. A case study is presented that supports the concept's introduction, and throughout the report the case study is revisited, providing consistent examples based on project experience.

There are various reasons why inefficiencies are encountered in the software lifecycle. Some of these can be controlled to a certain extent. There are several reasons, however, over which there is no control at all, or in some cases the symptoms for certain problems are difficult to trace.

There is nothing wrong with a "not so perfect" implementation of a change to a certain system (computer or organizational) as long as the business needs are fulfilled and the potential or emerging tradeoffs that pose some sort of constraint on the business capabilities are accepted. It becomes something wrong when compromises made during that implementation are forgotten. Constraints posed locally tend to propagate unnoticed and out of control and "infect" other parts of the system. Such hidden problems will strike during the next change that is made to that same piece of the system (or the infected pieces). If not corrected that time, it will strike even stronger during the next endeavor. The costs of fixing it tend to grow exponentially with each piece of the system that is infected and with each new layer of patches.

THE PROPOSED SOLUTION

The approach to managing such temporalities is based on three basic concepts:

1. **Temporarily acceptable solution** — accounts for the pieces of the IS environment where some constraining factor or discovered inefficiency has been located, usually resulting from factors external to IT

2. **Refactoring need** — should be considered as a recognized method for healing some (recognized or not yet uncovered) temporarily acceptable solutions, partially or in their entirety

3. **Refactoring needs map** — embodies the relations among temporarily acceptable solutions, refactoring needs, and elements of enterprise architecture

The concepts of the temporarily acceptable solution, refactoring need, and refactoring needs map can be utilized in the following areas related to technology management:

- **Enterprise architecture management** — scoping of architecture work, development of architecture changes, assessing architecture

- **Project scope elaboration** — development of project's feasibility study, assessment of technical risk included in solution under study, project tradeoffs valuation

- **Release management** — determining schedule for changes to be implemented, assessing work to be done to fulfill the requirements, synchronizing changes' deliveries

- **Software development** — detection of temporalities or constraints and their registration, determination of optimal resolution for temporalities, optimization of refactoring effort

CONCLUSION

Various benefits can be achieved by accepting such an approach to recognizing and handling temporalities. Due to the visibility and comprehension of the complexity rooted in technology and its utilization, the level of trust between IT and businesspeople tends to grow. Registering temporarily acceptable solutions and refactoring needs builds awareness of technical risks and constraints embedded in the IS environment and gives some idea of the implementation problems that must be addressed. This in turn gives more confidence to budget planning and work scheduling or leveling. In conjunction with refactoring needs, enterprise architecture brings more value not only to big IT-enabled projects being implemented but to the change management discipline as well. In combination with agile methods applied to software development, it is possible to achieve much wiser control over the quality of software being developed.

The concepts' acceptance among business representatives is usually amazingly easy. The way of thinking about temporalities is quite natural for them, as they've gotten used to dealing with such circumstances in their everyday work — today's business is founded on temporalities. Some difficulties, however, could be experienced during the start of the capturing processes; it could be difficult to convince IT people to spend a bit of time doing something they perceive to be purely administrative work.

The report presents the possible and validated way of having agile development processes efficiently coexisting and cooperating with the enterprise architecture and other technology management processes being utilized in the organization.