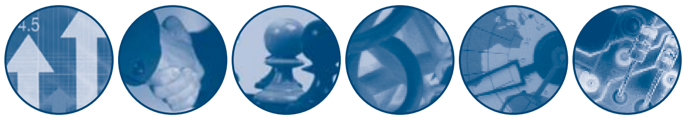

Update

Enterprise Architecture Advisory Service
Executive Update Vol. 12, No. 7

Don't Blame It All on Release Management

by Sebastian Konkol, Senior Consultant, Cutter Consortium

After the publication of Part I of my two-part *Executive Report* series[1, 2] on release management, I received some comments. Some of the issues mentioned could be seen as symptomatic of each organization that deals with release management. In light of these comments and from a recent consulting project at a mobile operator struggling with release issues related to its IT and VAS platforms environments, I decided to revisit release management and provide another perspective in this *Executive Update*: how release management helps uncover problems that can directly be attributed to software development, architecture, and collaboration.

SOURCE OF ALL EVIL?

Quite often I hear how bad release management is, as well as the hurdle it adds to technology management and operations. It is look at as the source of many problems and a major showstopper for valuable endeavors. The problems could be expressed in the following ways: "The releasing schema has such time constraints that we cannot deliver information systems of acceptable quality"; or "This requires much design up front — we all should be agile, but release management forbids us to do so"; or "Because of this release management, our IT environment transforms gradually into a hardly manageable, singular entity." Do you agree with such proclamations?

Let me start with some examples of problems with definitions (see Table 1), all said to result from release management. In my opinion, these problems have other sources. Having release management in place allows us to see these problems, but release management is not the true source. In addition to those listed in the table, numerous other problems are attributed to release management only because they are spotted in the context of release management tasks and duties.

Table 1 — Problems vs. Real Issues

The Problem	The Real Issue
Not standardized packages; contains modifications not related to architecture components.	Software development practices are not integrated with architecture management.
Modifications, including bug fixes, may be introduced only incrementally.	Software development practices are capable of fixing bugs only in patching mode.
Modifications are not ready to be validated on a gate, making the process late.	There is a lack of agility in the software development methodology being exercised.
Release scope regards the whole IT environment.	Architecture definition does not cover manageability and extensibility viewpoints.
Release environment buildup is very time-consuming.	Architecture definition does not cover manageability and extensibility viewpoints.
Validation on each gate takes too much time and effort (related to scope of changes introduced).	Scope of tests (especially regression) can't be limited due to lack of integration with architecture work.
Expected pace of modifications introduction is too fast to be controlled or to deliver quality software.	There is a lack of agility in the software development methodology being exercised.

FOR WHAT SHOULD WE AIM?

As identifying each problem represents potential for overall improvement, I am usually far from starting a religious war about whether release management helps. Still, let me present my view about what should be done with the real sources of the problems identified in the scope of release management. For example, Table 2 presents the problems identified and the solutions proposed within the project mentioned earlier. All problems noted were previously attributed to release management due to their symptoms observed in that context.

Release Management or Software Development?

So what problems in software development practice raise issues observed in the context of implementing releases? First, consider people, who are sometimes lazy and incompetent. Simply speaking, it is not enough for a developer to know the syntax of a programming language. He or she must also understand the purpose. Good programmers know their workshop, use patterns, and accept architectural constraints. I believe most of the problems in this scope are related to human nature. Apart from the discipline, I find a lack of standards to

Table 2 — Problems and Proposed Resolutions

Problems Identified		Resolution Proposed
Release Management	• Business releases and technical releases mashed up; business releases depend on technology release schedules. • Cascading releases in the process in parallel; next release built on not-yet-stabilized results of previous release. • Lack of releasing paradigms: architecture and design rules, no refactoring allowed. • Lack of discipline; cutoff "not mandatory."	• Fewer decision gates; decisions more condensed. • Minimization of cascading releases; less parallelism in release implementation. • Architecture should determine border of release contents. • Distinguish business releases from technical and refactoring releases. • Open and honest relations to business long term.
Software Development	• Missing or problematic competencies; good designer, good programmer, good manager. • Software developed internally interrelated; no possibility of detaching some modification from release contents. • No standards for handling emulators for external systems in case they are missing in development environment. • Lack of discipline; separation of modifications developed; acceptance of architecture and design patterns.	• Change in focus — from development to integration. • Extending development responsibility — from software integration to production acceptance. • Usage of architecture patterns, cooperation with architects on the level of design.
Architecture	• Missing or problematic competencies; good and responsible architect. • Lack of release management perspective in architectural work. • Standards for systems interconnection missing. • Architectural work quality; dynamism of IT environment extensibility.	• Business-focused IT environment decomposition onto areas to be released independently of each other, but coordinated inside each of them. • Real architecture management; cooperation with business on the strategic level instead of sitting and waiting where it will take us.
Business Relations	• Problematic way of thinking; forcing needs with no regards for constraints. • Lack of trust for a long time.	• Education to change way of thinking; "What can we do with what we have?" instead of "What is the most ideal thing we could require?" • Open and honest relations to business; new opening for better relations and building trust.

be a strong factor resulting in many problems. Those standards concern coding rules, use of libraries, or other design patterns.

It does not make sense to have the software development methodology defined with no regard for the releasing schema; they must cooperate efficiently. In this regard, the more agility used in the software development method, the better the overall outcome.

Release Management or Architecture?

As with software development, many release management problems rooted in architecture have human factors. The role of architecture is commonly stressed, but sometimes architects treat their work as a periodic task, something that takes place only during a big development. In truth, architecture and an architect's active support are crucial for release implementation to be successful. The architecture should define the division of the overall IT environment into the pieces that are released independently and determine standards or patterns to be used during a design effort.

Active support from architecture management is a must. Such support should regard the design of the architecture with its release in mind and in terms of the presence of the architects in the context of release decisions.

Release Management or Transparency and Collaboration?

The class of problems most difficult to handle is related to the way business and IT cooperate. Too many times, I have seen examples of IT covering its mistakes, such as obvious design faults or the inability to deliver, with statements that are on the border of the truth. To be fair, businesspeople do the same while, for example, justifying the necessity of having this very complex system working perfectly 24/7. Both blame the other side for their own faults. Such escape from responsibility usually results in the inability to make local decisions supporting overall company goals.

These folks must talk to each other, share risks and concerns, and act with bilateral support in mind, not their own politics. Company business cannot suffer due to some personal attitudes or local hierarchical games.

REAL PROBLEMS WITH RELEASE MANAGEMENT

Yes, some problems really are rooted in the inefficiencies or inadequacies of release management practice. These concern influencing architecture and design in order to be able to remove some modifications from release contents or the ability to introduce "quick and dirty" releases to be cleaned during subsequent refactoring release. But most often, the real problem is the lack of discipline in implementing release management tasks or the lack of power to force the execution of the rules involved.

CONCLUSIONS

After distilling some problems from symptoms observed in the scope of release management tasks, it became clear why I needed to examine some of the real issues. Most of these problems should be cured in other areas of technology management. They say that what cannot be measured cannot be managed. I agree with that. I think it is better to have the possibility of seeing the problem — even in the context of release management — and consciously cure it than not to notice it at all. Release management creates the framework under which problems encountered can be identified and properly addressed. The drawback is that, initially, release management will be said to cause problems — but it is still worth doing.

With respect to the problems presented, the relations among technology management processes can be illustrated in the way the cartoon character Shrek describes ogres: they have layers.[3] Problems not solved in cooperation between business and IT are the source of problems within architecture. In turn, unsolved problems in architecture raise issues in software development tasks, and, as we can see, all of them strike release management at the very end.

The true faults in release management are usually rooted in lack of discipline or lack of power. Necessary timing, decision-making efficiency, obeying of the governance rules, and appropriate integration with other technology management processes — these are the key lessons to be learned by the release management team.

ENDNOTES

[1]Konkol, Sebastian. "Release Management Framework: Part I." Cutter Consortium Enterprise Architecture *Executive Report*, Vol. 11, No. 11, 2008.

[2]Konkol, Sebastian. "Release Management Framework: Part II." Cutter Consortium Enterprise Architecture *Executive Report*, Vol. 12, No. 2, 2009.

[3]In the animated motion picture *Shrek*, Shrek tells his friend Donkey that "Ogres are like onions.... Onions have layers. Ogres have layers."

ABOUT THE AUTHOR

Sebastian Konkol is a Senior Consultant with Cutter Consortium's Enterprise Architecture and Enterprise Risk Management & Governance practices and is an enterprise startup and strategy consultant in Poland. He specializes in strategic IT planning, business technology partnership from both the technology and organizational perspectives, and fostering new technology management methods supported by social psychology. He is codeveloper of the Enterprise Architecture Management (EAM) approach and an expert in IT governance. He divides his time among running strategic IT consulting endeavors, managing projects, and sharing his expertise in EA management and IT governance. Currently, Mr. Konkol is developing strategic technology management frameworks that bind company strategy, EA, and corporate governance. As a social networking analyst and complexity theory enthusiast, he is also working on concepts that apply contemporary social psychology tools to the field of technology management. Mr. Konkol's professional goal is to help technology-dependent organizations achieve optimal technology usage in order to create and sustain competitive advantage, including harnessing the technical, business, and organizational perspectives through a partnership approach. Prior to becoming an independent consultant, in 2003, he worked, since 1997, for a mobile telecom operator as project leader in the field of GSM network surveillance and later as program manager responsible for running projects aimed at creating and providing GSM value-added services to the mass market. He has written numerous publications dealing with the business use of technology, telecom information environment management, and IT organization management. Mr. Konkol can be reached at skonkol@cutter.com.