Published In Proceedings of the 14th Annual Workshop of HP Software University Association, 2007

Building Blocks for Release Management Unification in Large-Scale Environments

Michael Brenner[1], Silvia Knittl[1], and Karin Schmelz (née Betz)[2]

[1] Munich Network Management Team
brenner@mnm-team.org, knittl@mnm-team.org
[2] HVB Information Services
Karin.Betz@hvbis.com

Abstract. In large organizations, Release Management practices have often evolved in organizational silos, resulting in a less efficient and riskier introduction of changes into the live environment. This work is based on an analysis of existing Release Management processes at HVBInfo. It highlights problems of uncoordinated release processes, challenges of unifying them, and basic building blocks for a comprehensive, ITIL-based Release Management approach.

1 Introduction

If anywhere then in enterprise IT infrastructures, there is nothing permanent except change. But uncoordinated, unplanned or insufficiently tested changes are a leading cause of IT failures [6]. From an IT Service Management (ITSM) perspective, it is Release Management that is responsible for introducing changes into the live environment in a controlled manner (see e.g ITIL [5] or Microsoft Operations Framework [4]). In Software Engineering (SE), usually a different view is taken, where a release is primarily regarded as a particular version of a single software product and Release Management as a project management task for handling the software provision within scheduled time and budget (see e.g. Rational Unified Process [3]).

The next Section introduces Release Management practices at HVB Informations-Verarbeitungs-GmbH (HVBInfo), where three distinct Release Management processes have evolved over time, resulting in various quality issues. Section 2 presents the basic approach for dealing with this situation, namely Release Management unification, its challenges and building blocks for addressing them. Section 3 discusses related work and section 4 concludes the article.

1.1 Scenario

HVBInfo is a service provider for the HypoVereinsbank (HVB) Group, supplying IT services like desktop services or hosting. The most important customers are the business units of HVB Bank. HVBInfo is running over 3400 servers, 30000 Clients and a number of IBM mainframes. New software for meeting the changing demands of the customers is developed either by external software producers or by HVBSystems, an application development organization within the HVBGroup.

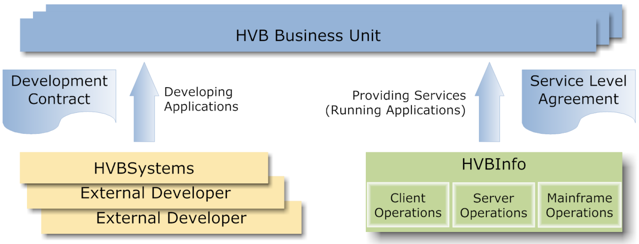

Fig. 1. Organizational Context

Note that there are contracts or agreements between the development organizations and the HVB business units, and Service Level Agreements underpinning the relationship between the business units and the operations units of HVBInfo – there is however no such formal agreement between any of the development and operations units. This situation (see Figure 1), where customers have separate contracts with the software producers and the service providers, is not unusual [7, 1].

At HVBInfo Release Management has evolved independently in each operations unit, resulting to platform specific release practices. There are three distinct processes for *Client/Server* (C/S – Client Operations), *zentrale Server* (zS – Server Operations) and *Host* (Mainframe Operations). While historically there might have been good reasons for this, the ever increasing interdependencies of components in the now significantly more networked infrastructure lead to a higher impact of the drawbacks of this approach.

1.2 The Trouble with non-unified Release Management Approaches

Within the scope of the described scenario different problem areas have been identified.
Issue A: *No common understanding of terms and concepts.* There are different understandings of the definition, the scope and the roles of Release Management between HVBInfo and the development organizations as well as within HVBInfo: The C/S Release Management has more ITSM-oriented characteristics whereas in zS, Release Management is mostly handled from a SE perspective.
Issue B: *Insufficient flow of information between development and operations.* HVBInfo does not get notice of new software developments until the completed product is transferred to them, making pre-planing of tests and user training complex.
Issue C: *Inefficiencies in managing releases.* There are ambiguous responsibilities, different wordings, and different release and testing policies per platform. Table 1 shows an extract of different release types per platform and their properties. Host and C/S operations, both use the concept of Standard Releases, trying to bundle a maximum of changes into a regularly scheduled release. On the zS platform on the other hand, there is no concept of a Standard Release. The concept of an Application Release used instead, is handled like a classical

software engineering project, thus the software producer is responsible for the whole release process. Among other undesirable effects, this complexity makes it all but impossible to coordinate releases phases, resulting in a higher number of rollouts, and consequently more downtime in the live environment, than necessary.

	C/S	**zS**	**Host**
	Client Operations	Server Operations	Mainframe Ops.
Defined Release Types	Standard Release (SR), Sonderversorgung, Hotfix (HF), Securityfix	Securityfix, Core Release, Application Release (AR)	Standard Release (SR), Sonderversorgung, Hotfix (HF), Hot Release
SR Frequency	≈ 5 per customer p.a.	n.a.	Adjusted to C/S
SR Size	Up to 380 products	n.a.	≈ 4 products
SR Duration	17 to 21 weeks	n.a.	8 to 10 weeks
HF Duration	≈ 2 weeks	n.a.	1 day
HF Responsible	Release Manager C/S	n.a.	Product Responsible
AR Responsible	n.a.	Project Manager	n.a.

Table 1. Examples of release types and properties at HVBInfo

Issue D: *Ineffectiveness in the protection of the live environment.* The existence of separate, uncoordinated release processes can weaken the effectiveness of testing, thus putting the stability of the live environment at risk. This is exemplified by the problem of intercalated hotfixes, illustrated in Figure 2. In this example, a release consisting of two C/S products (which are to be updated from version 2 and 4 to versions 3 and 5 respectively) is build and assembled in C/S development. They are then tested together in a test environment, which contains copies of various products in the live environment – among them the current version 7 of a zS product. However, while this standard release goes through a lengthy testing process, an unforeseen issue (e.g. a newly discovered vulnerability) triggers the development of an urgent change to the server component in zS development. As this urgent change is being released as a hotfix, it goes through an expedited testing cycle. The configuration of the zS test environment is reflecting the situation in the live environment, but does not account for pending releases for other platforms.

Why is that a problem? Since a hotfix is always a riskier change, there is an awareness that at the moment of its introduction, the live environment needs to be closely monitored. However, problems might manifest themselves not at that time, but much later. If the hotfix has "overtaken" the standard release prepared by C/S, the rollout of the standard release will result in an untested combination of products in the live environment – undermining the effort of testing the standard release thoroughly and diligently.

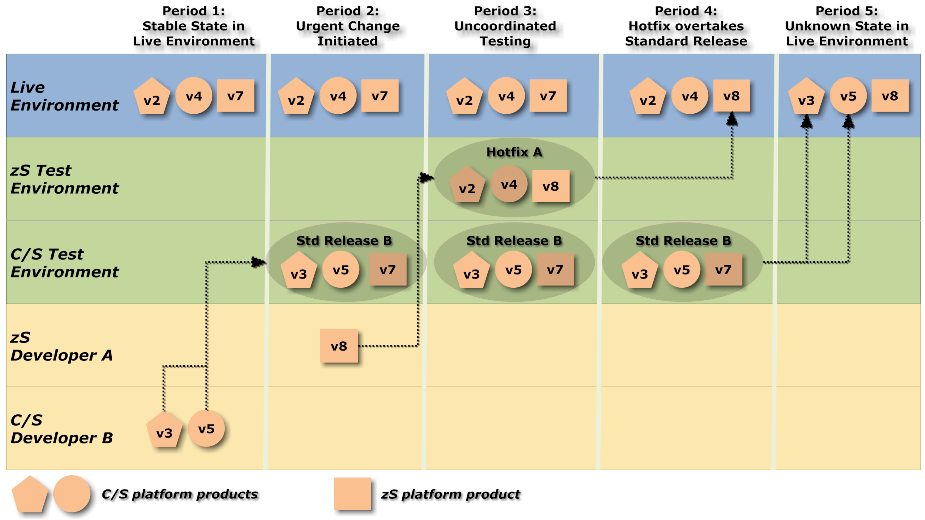
Period 1: Stable State in Live Environment
Period 2: Urgent Change Initiated
Period 3: Uncoordinated Testing
Period 4: Hotfix overtakes Standard Release
Period 5: Unknown State in Live Environment
Live Environment
zS Test Environment
C/S Test Environment
zS Developer A
C/S Developer B
v2
v4
v7
v8
v3
v5
Hotfix A
Std Release B
C/S platform products
zS platform product

Fig. 2. Intercalated Hotfix Problem

2 Solution – Unifying Release Management

This section describes, which challenges need to be addressed when unifying Release Management processes.

2.1 Challenges

RM Unification Challenge 1 (related to issue A) There should be a common theoretical foundation for Release Management – its scope, goals, terms and process – on which development and operations organizations can agree on.
RM Unification Challenge 2 (related to issue B) The "development-operations-wall". The lack of communication is mostly due to the the organizational constellation, which, as depicted in Figure 1, resembles an angle rather than a triangle. HVBInfo and the application development organizations interact mostly with their customers but not with each other. This is a rather typical situation: Even though the overall quality of the delivered IT services clearly depends on development as well as operations organizations, there is usually hardly any formal relationship or communication between them.
RM Unification Challenge 3 (related to issue C and D) The Release Management within HVBInfo needs to be unified. This involves several integration aspects, namely *roles*, *data*, *process* and *time*. New roles need to be introduced and existing roles need to be aligned and changed. Documentation requirements need to be harmonized, as well as process workflows. The frequency and timing of release cycles need to be adjusted in a way that is acceptable for the operations of all platforms (with regard to flexibility and stability concerns).

2.2 Building Blocks

There are three basic concepts needed for approaching the challenges stated in the previous section.

Addressing Challenge 1 This aspect is the topic of future work, for which the integration of Release Management practices from SE and ITSM standards into a single framework is planned.

Addressing Challenge 2 The"development-operations-wall" can be, at least partly, pulled down by an integration of Release Management with an ITIL-based Change Management process. In ITIL the difficult task of keeping a changing IT infrastructure stable is shared between Change Management and Release Management [5]. Figure 3 illustrates how in ITIL Release Management is integrated into the overall Change Management workflow (cp. also [4]). Basically, Change Management deals with the overall coordination of changes – from initial proposal, through authorization, up to a mandatory post implementation review. Releases are defined as collections of authorized changes that are to be tested and introduced into the infrastructure together. Consequently, most scheduling aspects as well as the task of change building, testing and implementing, are operationally delegated to Release Management, which controls release composition, testing and rollout. The introduction of an overarching Change Management can be the basis for a coordinated division of tasks and better communication between development and operations. Institutionalizing a Change Management process in which software changes are tracked from the moment they are being planned (i.e. submitted as a Request for Change), will enable HVBInfo to be be earlier and more consistently informed about future changes, thus making the planing of releases more predictable and stable.

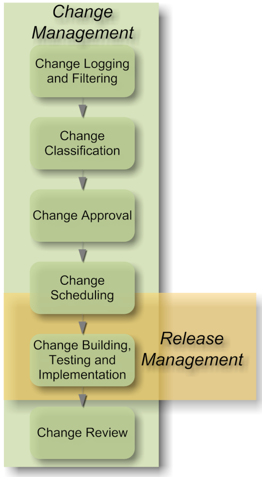

Fig. 3. Change & Release Management in ITIL

Addressing Challenge 3 To overcome the drawbacks of different Release Management processes rolling out components into a shared live environment, a concept for an integrated Release Management is proposed, which is platform-independent and therefore responsible for all changes. The new Release Management will have standardized process interfaces. New roles have been introduced, e.g. independent testers. By introducing this role, testing can be more flexible and its results more reliable. The test environment is now to be managed by one role, so that "overtaking" releases, as discussed in Section 1.2, cannot occur unnoticed anymore. Through better communication and coordination more changes can be bundled into a release, and thus the release frequency be lowered. Thus information and training of users and staff become more efficient.

3 Related Work

To the authors' knowledge, the are so far no publications specifically concerned with the practical issues of unifying Release Management in large enterprises. While overall the problem of integrating development and operations has not been researched nearly enough, some work is pointing in the right direction. MOF [4] includes a basic mapping of its review points to those of MSF, Microsoft's software development framework. CobiT [2] integrates practices for the entire IT lifecycle in a single framework, but its emphasis lies on control practices – its guidance for the development and operations processes themselves is not nearly as detailed as that of the dedicated SE or ITSM frameworks.

4 Conclusion and Outlook

The suggested steps towards a unified Release Management are currently under implementation at HVBInfo. The completion will take some time, as other urgent integration projects (due a recent merger of the parent company HVB) have currently taken precedence. Feedback by various stakeholders at HVBInfo was positive, which gives reason to hope that the presented building blocks for Release Management unification will find their way into the operational concepts of the future merged IT division.

The next step is to follow the progress of Release Management unification at HVBInfo and to review the resulting feedback. This should provide a good starting point for a further refinement of integration concepts. As the presented problems with non-unified Release Management plague many large enterprises, the proposed building blocks will hopefully find consideration in other scenarios, thus possibly laying the foundation for some kind of "best practices in integrated Enterprise Release Management". In the longer term, further research into the possibilities of integrating SE and ITSM frameworks (cp. *RM Unification Challenge 1*, Section 2.1) is intended.

References

1. M. Brenner, M. Garschhammer, and F. Nickl. Requirements Engineering und IT Service Management. In *Modellierung 2006*, volume 82 of *LNI*, pages 51–66. GI, 2006. `http://tinyurl.com/2vaz23`.
2. IT Governance Institute. *CobiT 4.0*, 2005. `http://tinyurl.com/24dhh3`.
3. I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison Wesley, 1999.
4. Microsoft Cooperation. *Microsoft Operations Framework – Service Management Functions – Release Management*, 2004. `http://tinyurl.com/29vp7e`.
5. Office of Government Commerce, editor. *Service Support*. IT Infrastructure Library. The Stationary Office, 2000.
6. D. Scott. Making smart investments to reduce unplanned downtime. Research Note TG-07-4033, Gartner, 1999.
7. R. Zarnekow, W. Brenner, and U. Pilgram. *Integriertes Informationsmanagement*. Springer, 2005.