Some Process Patterns for Enterprise Architecture Management

Christoph Moser[1], Stefan Junginger[1], Matthias Brückmann[2], Klaus-Manfred Schöne[2]

[1]BOC AG
Wipplingerstraße 1
A-1010 Vienna
{christoph.moser,stefan.junginger}@boc-group.com

[2]ZIVIT
Wilhelm-Fay-Straße 11
D-65936 Frankfurt
{matthias.brueckmann,klaus-manfred.schoene}@zivit.de

Abstract: The purpose of EAM patterns can be seen to supplement existing EAM frameworks, which often only provide generic approaches in implementing, maintaining, and analysing an Enterprise Architecture. This is done by providing additional concepts, tailorable to the EAM problems at hand. This paper extends the existing EAM patterns approach of the Technical University of Munich by defining a concept for EAM process patterns. Some EAM process patterns are presented in detail.

1. Introduction

Numerous frameworks for EAM are discussed in literature (for an overview see for example [Sc06]). Some of them are extremely complex regarding the number of concepts they consider. On the other hand, practitioners stress that only pragmatic and not too complex EAM approaches lead to a successful implementation within an organisation [Ke07]. This conforms to the experiences of the authors of this paper – two of us are with BOC, a company providing EAM solutions (ADOit, ADOben) and EAM consulting and two of us are enterprise architects at ZIVIT, one of the largest IT Service Providers in Germany's public administration.

We see EAM patterns as a powerful tool for implementing EAM within an organisation, especially for enhancing efficiency with the purpose of saving money and time. In the following we use the concepts presented in the EAM pattern catalogue of the Technical University of Munich, Germany [Bu08]. An important feature of the tool (ADOit) we use is its metamodelling capability [BO08]. Most ADOit users define their own EAM metamodel. These metamodels are derived from their EAM objectives and usually consist of only a few concepts (in contrast to complex metamodels described in many EAM frameworks).

In [Bu08] methodology, viewpoint and information model patterns are distinguished. Methodology patterns are derived from so-called concerns. When implementing EAM (with ADOit) we use a similar approach which is depicted in figure 1.

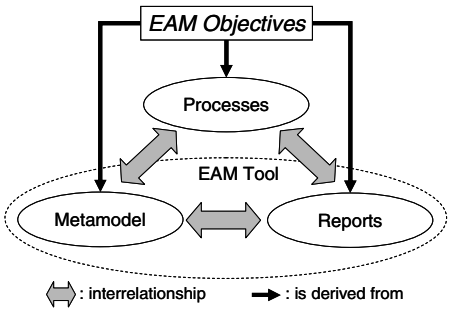

Figure 1: Approach for implementing EAM

Starting from the EAM objectives we define within an EAM implementation project a) the metamodel (sometimes also called data model or information model), b) the needed reports and c) the processes for implementation, maintenance, and utilisation of the EA. The element "Metamodel" maps to the information model concept of [Bu08], the element "Reports" maps to the viewpoint concept[1] and the element “Processes” is part of the methodology patterns. The EAM objectives correspond approximately to the concern concept described in [Bu08]. However, we see EAM objectives on a higher level: We distinguish EAM objectives such as documentation of the application landscape, consolidation of the application landscape (e.g. by removing redundant applications or harmonising the application landscapes of different locations/countries), consolidation of hard- and software and SOA governance. Of course, these EAM objectives overlap (and so do the patterns to achieve them).

An important element of EAM approaches not considered in detail within the methodology patterns of [Bu08] is the processes.[2] By "Processes" we mean the processes that define how EAM is done (not the business processes which are supported by IT). In our EAM implementation projects we experienced that there is a set of recurring "process patterns", e.g. different patterns how an EA repository can be built up and kept up-to-date, which roles shall be responsible for performing which tasks. It has to be noted, that some EAM frameworks (like TOGAF [TO06] and FEAF [CI99]) provide generic procedural models for implementing EAM. However, these procedural models can only be used as a starting point because of their high level of abstraction. In contrary, we see EAM processes on a much more detailed level. Akin to the support of service management processes in ITIL by a Configuration Management System (CMS) [OG07], EAM processes might be supported by an EAM tool (in this case, the EAM tool ellipse in figure 1 needs to be extended).

[1] For a broader discussion on the concepts of concern, view and viewpoint refer to [AI00] and [Sc04].
[2] However, in [Er08] improvement potential for the current version of the EAM pattern catalogue is discussed. It is outlined, that besides additional improvements, the methodology patterns will contain more structured process descriptions in the upcoming version of the EAM pattern catalogue.

The remainder of the paper is organised as follows. In chapter 2 EAM processes are discussed, a definition of EAM process patterns is given and the pattern language we use to describe EAM process patterns is presented. Chapter 3 presents concrete EAM process patterns. The interplay and usage of these patterns is discussed, by integrating these with EA metamodels and EA reporting patterns. Finally, chapter 4 gives an outlook on future research fields. To illustrate the concepts presented in this paper we refer to the EAM implementation at ZIVIT. Details about this implementation can be found in [Ju08].

2. A Pattern Form for EAM Process Patterns

There is not much literature about EAM processes (in comparison to publications about EAM modelling frameworks). Of course, the EAM processes needed depend on the EAM objectives. If, for example, the EAM objective is to standardise soft- and hardware a process is needed on how this standardisation should take place. Additionally, it has to be decided which processes are seen as part of EAM as it possesses – and of course has to possess – a tight integration with disciplines such as business process management, strategy management, (IT) service management, demand management, (project) portfolio management, requirements management, software development, risk and compliance management. Hence, some organisations see demand and portfolio management as part of EAM and others do not.

To classify EAM process patterns concerning their utilisation within EAM processes, we use the high level EAM process landscape shown in figure 2. The process landscape is generic regarding EAM objectives and even abstracts from descriptions as they can be found for example in [De03], [TO06], [Ke07], [BO08], [Mo08].

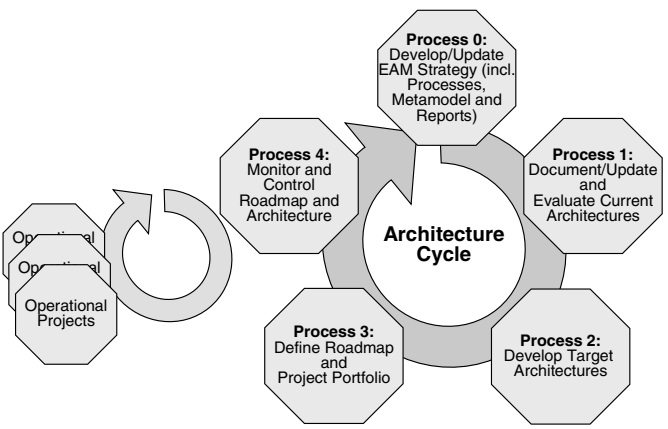

Figure 2: High level generic EAM process landscape

There is a separate research field dealing with process patterns. For example in the BPM/workflow area business process patterns are examined for many years [Aa03]. However, they usually focus on non-domain specific elements in business process models such as control structures. Therefore, the results from this research area do not fit

to our intention of EAM process patterns. In the software engineering community there are many publications about process patterns for the software development process [Co95]. Furthermore, concepts described in CMMI and ITIL can be seen as process patterns, too – although not described this way [OG07], [CM06]. ITIL states for example, that the main purpose of its CMS is the provision of up-to-date and secure information via the configuration items used to support all service management disciplines [OG07]. Hence, there is a major overlap with process patterns for keeping an EA repository up-to-date ("Process 1" in figure 2). The concepts developed in these fields fit better to our objectives – just the application area is different (EAM vs. software development/IT service management). We define an EAM process pattern as a "reusable element of an EAM process (model)". Usually, an EAM process pattern is parameterised in the sense that exact activities and roles executing the activities are defined when applying the pattern.

We describe EAM process patterns using the following "pattern form":

- **Name:** Concise, strong name for the pattern.
- **Summary:** Short description of the process pattern.
- **EAM process:** Description in which EAM process this pattern can be used. For this purpose we classify the patterns according to the process landscape shown in figure 2.
- **Problem:** Indication of the situation to which the pattern applies, and if applicable the entry conditions to perform it.
- **Solution:** Description of the process pattern including the steps/activities to be performed. We use BPMN to illustrate the patterns.
- **Resulting Context:** Description of the situation/context which will result from performing the process pattern solution.
- **Related Patterns:** Indication of patterns that this pattern is composed of, is a part of, or is associated to. To describe the associations to metamodel (information model) and reports (viewpoints), we refer exemplarily to EAM patterns described in [Bu08].
- **Known Uses/Examples:** Description where/how the process pattern has been applied.

3. EAM Process Patterns: Some Examples

This chapter identifies an extensible set of reoccurring patterns in EA processes and shows the best use practices for them. The patterns are derived from the practical experience of the authors in EAM and literature.

3.1 Pattern: Centralised Manual Data Acquisition/Maintenance

Summary: Architecture artefacts of a certain type (class) are maintained in the EA repository by a central role, usually the enterprise architects.

EAM Process: Process 1.

Problem: This process pattern can be applied if the EA repository shall be filled or updated.

Solution: A small group, usually enterprise architects, is manually maintaining the EA repository. However, normally even in small organisations the enterprise architects need input from experts of different domains. Depending on the used metamodel domain experts could be business process experts, application owners and ICT experts. The tasks of this pattern are depicted in figure 3.

If the architecture artefact is isolated it is simple to update the EA repository. However, usually architecture artefacts are strongly interrelated between each other. For example, an application and all interfaces offered by it usually build a logical entity. All artefacts of the logical entity need to be updated ideally within a small time span to avoid inconsistencies (like dangling interfaces when deleting an application). This reduces the time during which the EA repository is in a non-consistent state.

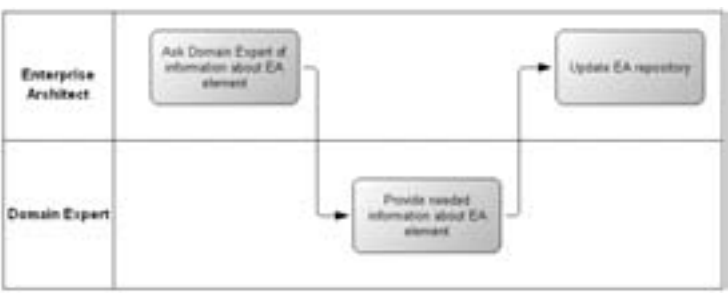

Figure 3: Tasks of the EAM process pattern "Centralised Data Acquisition/Maintenance"

Resulting Context: The EA repository is updated and in a consistent state. Data consistency and a logical structure of the architecture artefacts are granted.

Related Patterns: This pattern might be used in combination with the "release workflow pattern". The methodology pattern "Management of Homogeneity" (see [Bu08], M-21) is one example for using this pattern to catalogue the technologies within a centrally maintained structure.

Known Uses/Examples: The pattern applies especially for those architecture artefact types, where the required know how for structuring is not widespread within the organisation or an overall structure needs to be developed first.

An example is the definition of a hierarchically structured process landscape, usually defined by a small team of business experts. Another example is described in [Ju08]. In this case software, technology and hardware artefacts are centrally maintained by the enterprise architects (see figure 4). The enterprise architects – amongst other things – are responsible for provision of the mentioned artefacts in an organisation-wide agreed structure. Any application owner would use these artefacts for describing his application by assigning those architecture artefacts to his application. In the given scenario applications – in contrast to technologies, hardware and software – are maintained decentralised (see chapter 3.2).

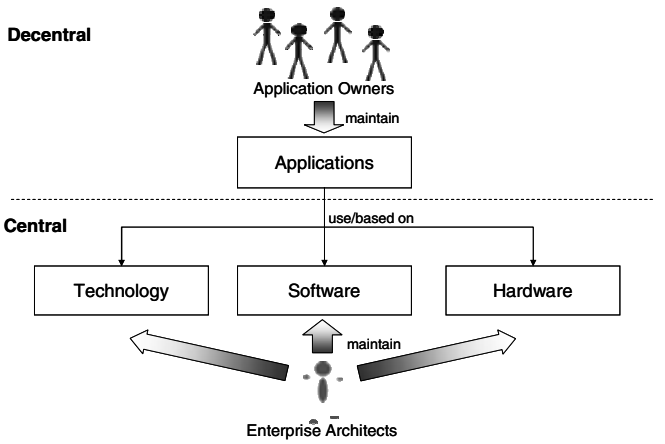

Figure 4: Example for centralised and decentralised maintenance of an EA repository

3.2 Pattern: Decentralised Manual Data Acquisition/Maintenance

Summary: Parts of the EA repository are manually maintained by domain experts.

EAM Process: Process 1.

Problem: This process pattern is used for defining and maintaining architecture artefacts for which a central maintenance is not feasible. This applies especially to architecture artefacts where the needed knowledge for documenting and analysing is widespread within the organisation; e.g. a large group of application owners.

Solution: The domain expert updates the EA repository for architecture artefacts he is responsible for. If required, he requests architecture artefacts from other domain experts or from a central role (usually the enterprise architects). For example software or hardware artefacts might only be defined by enterprise architects or used interfaces might be defined by other application owners. The tasks are depicted in figure 5.

To assure high quality of the contents of the EA repository changes to architecture artefacts should be stored in a change log. Within the change log it is recorded which changes have been done by whom and when. A more powerful way is to use the "release

workflow pattern" (see "Related Patterns"). Furthermore, the owners of architecture artefacts should confirm the accurateness of their data on a regular basis, to demonstrate the accurateness to the possible users (this could of course be seen as a separate pattern). It has to be noted that this process pattern implies interesting requirements (intuitive GUI, access rights, operation of the EAM tool etc.) to the EAM tool at hand.

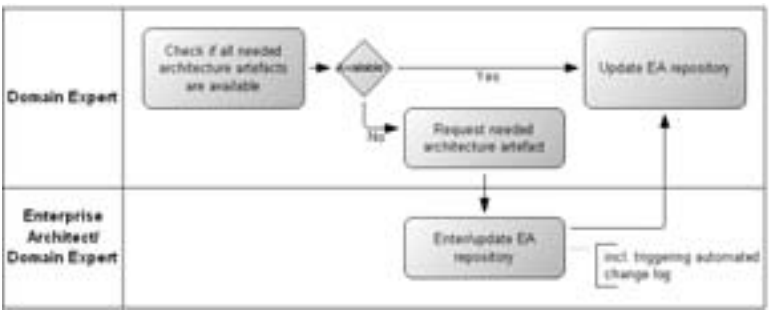

Figure 5: Tasks of the EAM process pattern "Decentralised Data Acquisition/Maintenance"

Resulting Context: A decentrally maintained architecture artefact is added or updated.

Related Patterns: This pattern might be used in combination with the "release workflow pattern". The methodology pattern "Management of Interfaces" (see [Bu08], M-21) requires assigning interfaces to applications. Typically a multitude of applications and interfaces need to be recorded/maintained. If these need to be kept up-to-date within an EA repository, it is nearly impossible to achieve this by a centralised maintenance.

Known Uses/Examples: Most organisations possess at least dozens of applications and there are usually hundreds of interfaces between them. In this case, a decentralised approach for maintaining the applications and belonging interfaces is preferable (see [Ju08] and figure 4).

3.3 Pattern: Automatic Data Acquisition/Maintenance

Summary: This process pattern allows for keeping the EA and corresponding architecture models automatically up-to-date and consistent by importing models not controlled by the EA architects.

EAM Process: Process 1

Problem: Often manual maintenance of the EA is not feasible. Reasons might be the low "willingness" of domain experts to provide the required data who are often forced to maintain multiple data sources, by providing nearly the same data, for example architectural models (like process architectures) which might not initially be developed to support the EAM initiative, but are valuable for EAM.

Solution: Precondition to this pattern are agreed upon data delivery contracts (comprising interface description to the source system, transformation rules, data quality etc.), as proposed by [Fi06]. A major challenge is to define the transformation rules to transform the delivered "external source models" into the required format given by the metamodel of the EA repository. Furthermore an adequate EAM tool needs to be in place, capable to import the data of multiple data sources. The proposed process pattern, depicted in figure 6, is based on the process for data maintenance discussed in [Fi06]. The source models need to be checked against the data delivery contract. If this quality check (content and data consistency) fails, the data provider is responsible for performing corrective measures and for delivering the data in the agreed quality. After passing the quality check the data is transformed and imported into the EA repository. The intended changes need to be evaluated by the various EA stakeholders before being propagated into the released part of the EA. Helpful mechanisms such as logical deletion of architecture artefacts, means for baselining and multi-dimensional versioning are discussed in [Mo08].

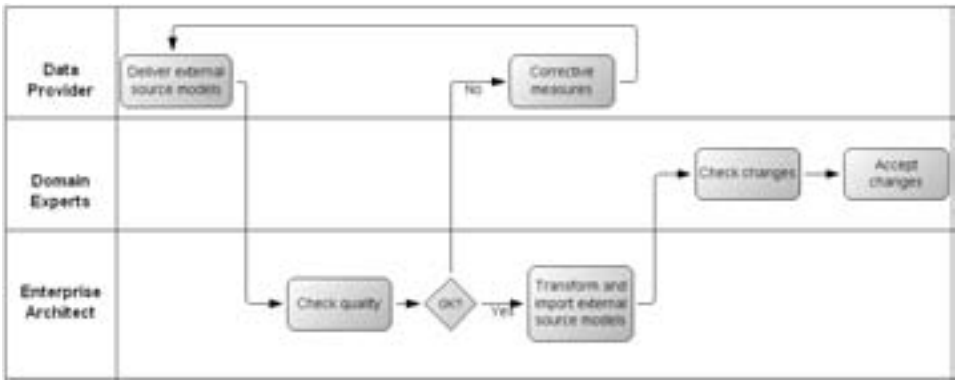

Figure 6: Tasks of the EAM process pattern "Automatic Data Acquisition/Maintenance"

Resulting Context: The EA is updated, by importing architecture artefacts from external sources.

Related Patterns: This pattern might be used in combination with the "release workflow pattern". The Release Workflow might be used to accept automatically updated architecture artefacts. The methodology pattern "Analysis of the Application Landscape" (see [Bu08], M-13) determines which business processes are supported by which applications. To avoid multiple data acquisition efforts process landscapes might be imported from a BPA/BPM tool.

Known Uses/Examples: Practical experiences are described for example in [Fi06].

3.4 Pattern: Architecture Control by Applying a Release Workflow

Summary: The objective of applying a Release Workflow is to ensure that only authorised and identifiable architecture artefacts are recorded within the EA repository.

EAM Process: Process 1, Process 2, Process 3, Process 4.

Problem: Especially if the "Decentralised Manual Data Acquisition/Maintenance" is applied there is the danger that the data in the EA repository is not accurate. This process pattern tries to overcome this by allowing only authorised changes of architecture artefacts.

Solution: If architecture artefacts are to be added, modified, replaced or removed the RWF pattern can be applied, see figure 7. [Mo08] propose a RWF realised as a status automaton that defines the states "Draft", "Audit", "Released" and "Archived". Changes are drafted (by defining a new versioning branch) and submitted for acceptance (without changing the released EA). When approved (usually by the enterprise architects in accordance with predefined architecture principles, see [TO06]), the new version is released into the EA and the old version is archived.

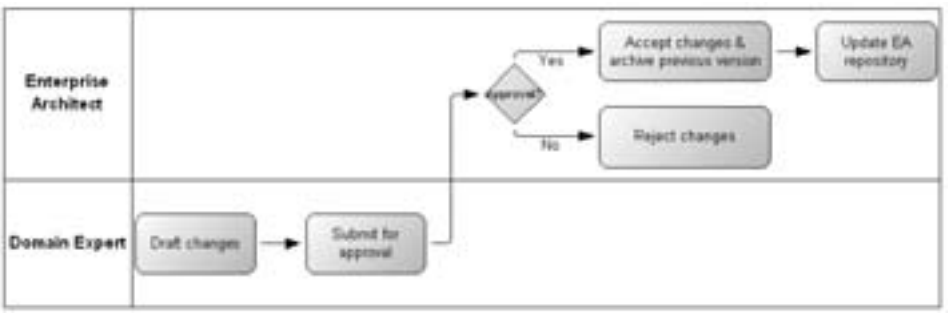

Figure 7: Tasks of the EAM process pattern "Release Workflow"

Resulting Context: The EA repository is updated and in a consistent state. Changes have been approved after adequate quality assurance.

Related Patterns: All process patterns for data acquisition/maintenance might be combined with this process pattern.

Known Uses/Examples: In [Ju08] the RWF is applied to architecture artefacts of the type "Application". Changes to applications are performed decentrally by the responsible application owners. Hence, they need to be authorised by enterprise architects.

3.5 Pattern: Lifecycle Management

Summary: The objective is to evaluate the state of architecture artefacts and to pick retirement candidates, to keep the EA as consolidated as possible.

EAM Process: Process 1, Process 4.

Problem: To control architecture artefacts efficiently during their entire lifespan, it is necessary to assign lifecycle states to architecture artefacts. By assigning lifecycle states to architecture artefacts stakeholders are guided in their architectural work. Planning activities are supported and development of the EA in accordance with the technology roadmap – comprising the agreed set of software, hardware and technologies to be used – of the organisation is possible.

Solution: Prerequisite to lifecycle management is that for each type of architecture artefact (modelling class within the metamodel) the possible lifecycle states are defined. Furthermore conditions to transfer an architecture artefact from one state into another need to be defined. [Mo08] for example propose the states "Planned", "In Test", "Implemented", "In Phase-out" and "Retired". Once the state "In Phase-out" is assigned to an architectural artefact planning to manage the possible migration to a new target artefact is triggered, see figure 8.

Figure 8: Tasks of the EAM process pattern "Lifecycle Management"

Resulting Context: All architecture artefacts feature a lifecycle state. Monitoring of the EA is possible and tracking/analysis of architecture roadmap compliance is possible.

Related Patterns: For the assignment of lifecycle states any of the aforementioned data acquisition/maintenance process patterns might be used. An example for applying this process pattern is the methodology pattern for reducing the heterogeneity of the technologies of the application landscape (see [Bu08], M-3). This process pattern might be applied to assign the lifecycle states to the technologies in use.

Known Uses/Examples: [Ju08] discusses lifecycle management for architecture artefacts of the type software, hardware and technology. For the given problem the lifecycle of these artefacts is monitored with the goal of saving maintenance costs. To avoid running into more costly extended maintenance or even supreme maintenance, these artefacts are centrally monitored. By applying adequate reports (see [Ju08] for details) necessary initiatives after a status change (e.g. setting up a project to adapt applications affected by a status change of their underlying database management systems) can be derived.

3.6 Pattern: Verification and Audit

Summary: Audits confirm that the artefacts of the EA conform to the agreed standard and verify, if the current situation (real world) reflects the details in the EA repository.

EAM process: Process 1, Process 4.

Problem: This pattern can be applied to ensure consistency and actuality of the EA. It is of major importance to assure that the EA repository, providing the basis for any architectural work is up-to-date because otherwise planning deficiencies, caused by utilisation of inconsistent and outdated data might appear.

Solution: [CM06] differentiates various types of audits for keeping a CMS accurate. Most important for EAM appear a) Documentation Audits, conducted to confirm that EA records and architecture artefacts are complete, consistent, and accurate and b) Physical Audits, conducted to verify that the current architecture (real world) conforms to the technical documentation that defines it. Possible deviances need to be eliminated. Furthermore it is to be checked, whether identified deviances affect current initiatives. If necessary, corrective measures need to be triggered. Verification and Audit is usually performed by the enterprise architects. Figure 9 depicts the pattern.

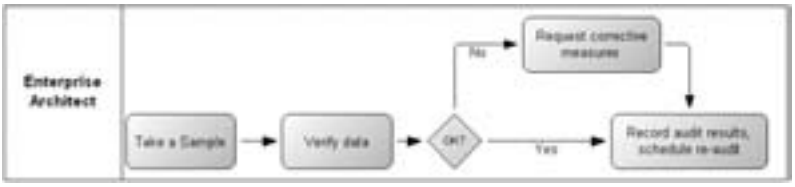

Figure 9: Tasks of the EAM process pattern "Verification and Audit"

Resulting Context: Inconsistencies and bad quality are uncovered and corrective measures are triggered.

Related Patterns: The aforementioned process patterns for data acquisition/maintenance might be triggered if "Verification and Audit" uncovers any inconsistencies.

Known Uses/Examples: Verification and Audit are common practice in the fields of Software Engineering and Service Management.

4. Closing Remarks

Although already implemented in some organisations, EAM is not yet a commodity [Jo06], [Ke07]. One reason is of course that it is difficult to calculate a ROI and the realisation of the benefits of EAM needs some time (usually years). Additionally, in our opinion the successful implementation of EAM mainly depends on a successful implementation of the EAM processes. Finally, the EAM processes are depending on the EAM objectives of the organisation and their integration with processes of other disciplines such as definition/update of the IT strategy, portfolio management, project management etc. Therefore, it is difficult to find an appropriate level to do research about EAM processes and exchange experiences. We see EAM process patterns as a powerful tool to do so. Therefore, we would appreciate a research community dealing with EAM process patterns. The patterns presented in this paper might be a first step towards this vision. The presented patterns cover some aspects of maintaining an EA repository. Of course, there are many more patterns for other EAM processes. In

addition to the patterns themselves the used pattern forms need to be reviewed and probably refined.

5. References

[Aa03] van der Aalst, W.M.P. et al.: Workflow Patterns. In: Distributed and Parallel Databases, 14(1), 2003, pp. 5-51.

[AI00] ANSI/IEEE Std 1471-2000; Recommended Practice for Architectural Description of Software-Intensive Systems.

[BO08] BOC AG: Enterprise Architecture Management with ADO*it*. Wien, 2008.

[Bu08] Buckl, S. et. al.: Enterprise Architecture Management Pattern Catalog. Release 1.0, Garching b. München, Germany 2008, http://srvmatthes8.informatik.tu-muenchen.de:8083/file/EAMPatternCatalogV1.0.pdf (access: 2008-10-15).

[CI99] CIO Council: Federal Enterprise Architecture Framework. Version 1.1, http://www.cio.gov/Documents/fedarch1.pdf (access: 2008-10-15).

[CM06] CMMI Product Team: CMMI® for Development. Version 1.2 (CMMI-SE/SW/IPPD/SS, V1.1): Staged Representation, Carnegie Mellon Software Engineering Institute, http://www.sei.cmu.edu/cmmi/models/index.html (access: 2007-08-15).

[Co95] Coplien, J.O.: A Generative Development-Process Pattern Language. In: Coplien, J. O.; Schmidt, D. C. (Eds.): Pattern Languages of Program Design. Addison Wesley. 1995, pp. 183-237.

[De03] Dern, G.: Management von IT-Architekturen. Vieweg, Wiesbaden, 2003.

[Er08] Ernst, A.: Enterprise Architecture Management Patterns. http://www.hillside.net/plop/2008/ACM/ConferenceProceedings/papers/PLoP2008_18_Ernst.pdf (access: 2009-03-01).

[Fi06] Fischer, R. et. al.: A Federated Approach to Enterprise Architecture Model Maintenance. In: Reichert, M. et al. (Eds.): Enterprise Modelling and Information Systems Architectures, LNI P-119, GI, Bonn, 2007, pp. 9-22.

[Jo06] Jonkers, H. et al.: Enterprise Architecture: Management tool and blueprint for the organisation. In: Information Systems Frontier (2006) 8, pp. 63-66.

[Ju08] Junginger, S. et. al.: Anwendungsportfoliomanagement mit ADO*it* im ZIVIT. In: Riempp, G.; Stahringer, S.: HDM-Praxis der Wirtschaftsinformatik: Unternehmensarchitekturen. dpunkt-verlag GmbH, 262, 2008, pp. 29-38.

[Ke07] Keller, W.: IT-Unternehmensarchitektur. Von der Geschäftsstrategie zur optimalen IT-Unterstützung. dpunkt.verlag, Heidelberg, 2007.

[Mo08] Moser, C. et. al.: Business Objectives Compliance Framework. Mechanisms for Controlling Enterprise Artefacts. In: Kühne, T.; Reisig, W.; Steimann, F. (Eds.): Modellierung 2008. Proceedings, March 2008, Lecture Notes in Informatics, Volume P-127, pp. 74-88.

[OG07] Office of Government Commerce: ITIL – Service Transition. Appeared in the book series ITIL - IT Infrastructure Library, The Stationery Office, London, 2007.

[Sc04] Schekkerman, J.: Another View on Extended Enterprise Architecture Viewpoints. http://www.enterprise-architecture.info/Images/Extended%20Enterprise/E2A-Viewpoints_IFEAD.PDF (access: 2008-10-15).

[Sc06] Schekkerman, J.: How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an EA Framework. Trafford Publishing, 2006.

[TO06] TOGAF: The Open Group Architecture Framework, Enterprise Edition. Version 9, http://www.opengroup.org/architecture/togaf9-doc/arch/ (access: 2009-02-15).

[Za87] Zachman, J.: A framework for information systems architecture. In: IBM Systems Journal, 26(3), 1987, pp. 277-293.