Meta-Architecture Tools: the Working Engines of the Company's Architectural Evolution Practitioner Position

Ethan Hadar[1], Irit Hadar[2]

[1]. Central Architecture , Mercury Interactive
19 Shabazi Street, POB 170, Yehud 56100 Israel
+972-3-5399524
ehadar@mercury.com

[2]. Department of Management Information Systems
University of Haifa
Carmel Mountain, Haifa, Israel
+972-4-8288508
hadari@mis.haifa.ac.il

Abstract. The need to facilitate architectural centric evolution (ACE) of enterprise products is a risky mission. It involves different methodologies of the existing products and diverse best practices. Strong and efficient communication must be facilitated between the product lines with accurate, comprehensive and rapid peer reviews.
Regular reviews are the crucial engine for successful architectural evolution. However, since the product lines are constructed from different technologies, communities, experience gained and more, a need for commonality and mutual understanding is crucial.
This paper presents the Meta-Architecture tools constructed at Mercury Interactive Company as the main engines to assist architects in performing their tasks. The tools are implemented within: (1) the Mercury Modeling community; (2) the Mercury Modeling Standard; and (3) the Mercury Architecture and Design Lifecycle.
In this paper we present the rationale and actual implementations of these tools and the corresponding method of operations, as well as its challenges and solutions.

Keywords: Architectural Centric Evolution, Meta Architecture, Composition Refactoring, Software Process, Design Reviews, UML

1 Introduction

The ongoing architectural centric evolution (ACE) [6] of large scale products is known to be the driving engine of major development problems. Realization of these responsibilities, when assigned to the team's architects may by incorporated into activities such as (1) defining and changing components, responsibilities and services; (2) defining subsystems by means of product decomposition and layers selections; (3) marinating the product versioning by expansion or complex refactoring [3]; (4) integrating and wiring to external systems; and many more. The description of the current architectural situation, and the proposed high level solution, need to be communicated and constantly verified against cross-organizational concerns. Understanding the nature of these verifications, the company must adapt its methods of software process engineering to the architectural centric ones, and therefore must assist the architects and development teams by creating tools and means to foster their ACE agenda.

However, since the level of experience is not unified among all research and development organization, one cannot assume that the mere creation and representation of architectural views is appropriate and sufficient. Consider the case of impact analysis in which the management needs to quickly asses the consolidated impacts of changes within cross organizational constrains. Such a case, when related to cross products architectural issues regarding products integrations, can become an architect's nightmare. In some cases, it may even be the consequence of simple misunderstanding or difference of opinions.

Central Meta-Architecture tools are the common building blocks used by the organization's architects and R&D teams, when implementing the ACE needs. It is materialized as interweaved collection of methodologies, skill sets, check lists, guidelines, common notations, and unified design and review lifecycles. They are supported by an implementation of UML profile and its relevant implementation within a CASE tool with specific company defined type of diagrams. These artifacts should assist in creating common understanding, enable quick decision making, and generate focus on the problem at hand. The usage of these tools should be understandable not only by architects, but rather by other, non-architects, members of the team, by means of efficient knowledge assimilation. These tools implementation generates a roadmap for training new architects in the company, as well as contractual templates between the teams' designers and reviewers.

This paper presents practical implementations of such Meta-Architectural tools as implemented by a Central Architecture team. It enabled the company to consolidate models from different products lines into a single model, with the same notation and information structure, in order to define and plan the next architectural centric evolution of the products separately and the common company infrastructure specifically.

The next section describes the relation between Meta-Architecture and ACE and the three pillars of Meta-Architecture, followed by their respective building blocks presented and demonstrated in sections 3-5. Section 6 concludes the theoretical model in light of its practical demonstration.

2 Method of Creation of a Meta-Architecture Tool

The realization of these Meta-Architecture artifacts and methodologies emerged according to interviews and observations conducted by the company's technological and management executives. It involved the perspective of both team leaders and individual developers on the Meta-Architecture ideology and its proposed implementation, and is constantly improved and modified according to changing market needs.

Each artifact and methodology is initially proposed by the Central Architecture team or by the divisions' architects, and is adapted in the company level. In order to understand the difficulties of such tasks and the need for constant evangelism, one must realize the size of the relevant R&D division. The company's R&D software engineering organization employs over 750 developers, divided into more than 120 teams in 11 product lines. It includes many different technological tools such as various programming languages and environments (e.g. JAVA, C++, C#, script languages, J2EE, .NET), operating systems, connectivity over numerous protocols, and more. The ecosystem of such company is characterized by its diversity, and the phenomena of NIH ("Not Invented Here"). Therefore each Meta-Architecture tool that is being developed must be customizable by its nature, evolve and improve its efficiency during the ongoing development of each of the products.

The driving force for each tool creation needs to be concentrated around the three major ACE business objectives (see figure 1):

(1) to communicate and provide a coherent agreement on
(2) design decisions, as early as possible, based on a
(3) simple abstract representation of current and proposed architecture

To this aim, meta-architectural tools need to construct solutions according to:

(1) providing common understanding regarding the methods of software creation, by means of
(2) representing architectural views in a unified notation using common CASE tools, while maintaining
(3) ongoing flow of design and review sessions, based on simple representations.

The next sections elaborate the three elements of Meta-Architecture as applied at Mercury: the Mercury Modeling community; the Mercury Modeling Standard; and the Mercury Architecture and Design Lifecycle.

3. Mercury Modeling Community

This pillar is the most important one, driving the motivation and construction of the other pillars. It concentrates on evangelism, educational roadmap, and the correlated professional levels and technological skills needed by different stakeholders. Simplicity is the main concept of inserting any practical design and architectural methodology. These methodologies are reflected by the results of modeled architecture's views. By maintaining the amount of different views and their respective level of details (conceptual, logical, and detailed), one can convey a considerable amount of information while keeping a good enough understanding of the architecture.

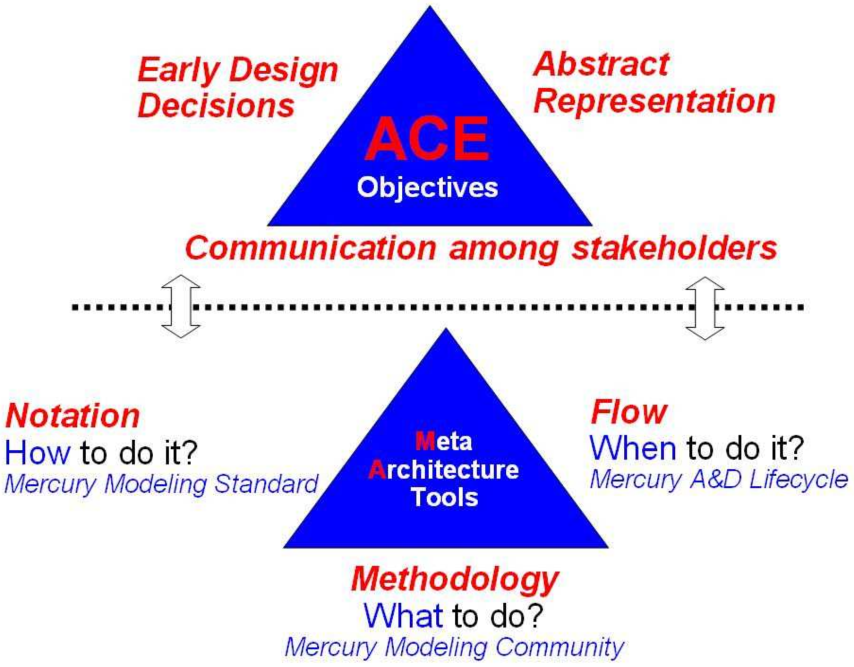

Fig. 1. : ACE business objectives mapped to Meta-Architecture tools

For this aim we tailored specific training roadmaps and educational pocketbooks, which match the stakeholders' responsibilities.

The emphasis in such training is leaning towards simple, practical, focused workshops at the company site, with homogenous teams as much as possible. These training involved the assimilation of education speedup artifacts. In what follows we present several examples.

3.1. Views Orientation Mapping

A lookup table which directs a business concern to architectural views and their respective level of details. Meaning, it connects direct needs to architectural solutions, without complex educational process. Consider the example of the business need: "what information do I need in order to conduct impact-analysis on the existing architecture?" The result will be "Use decomposition view with conceptual level for subsystems, logical level for high-level components, and detailed level for internal relations among classes (inner components diagram)". If the user needs further elaboration on the decomposition view, s/he is directed to the internal company's website for simple description on the content of the view. Moreover, the user is provided with a specific UML profile, adapted to the CASE tool, with additional usage explanations on how to generate the diagram and conduct the analysis.

3.2. UML Arrow Rules

This is a simple check list artifact, aimed to assist the designer in selecting the appropriate association between two classes. This tool consolidates all the information needed when considering the most problematic design task after defining responsibilities' division. Namely, association selection. Examples can be generalization vs., aggregation, reference vs. values, usage vs. reference passing and so on.

3.3. Specific Methodologies Training

The teams also conduct methodological training on how to change a complex system, and how to conduct practical implementation of design patterns. An example of such a methodological training is the implementation of Complex Refactoring [2] change, by means of collaborated risk estimation methodology, code investigation, and the change process [4].

4. Mercury Modeling Standard

This second major element is the practical tool for combining the implementation of the methodologies (the selection of views and their content) and design sessions and reviews (the third element). In order to facilitate proper reviews, the review preparation by the reviewers needs to be thorough, and the generated impact of the decision makers must be accurate. This means that the preparation and summation tasks are time consuming for both reviewers and management. However, in a commercial environment, when the reviewers are peers from other products or other teams, the time spent outside their own agenda is considered by their direct managers as overhead. Consequently, it may cause the review procedure to become the bottle neck of the development teams, or even, in a worst case scenario, a mere rubber stamp on the design.

Therefore, the goal of the Meta-Architecture team at Central Architecture is to minimize the time consuming preparation tasks, and maximize the impact understanding of the reviews. It is done by two major activities: (1) implementing common CASE tools and common notations of architectural views; and (2) defining minimal set of known and cross company communicated views.

The uniformity of "How to do it" can be tackled with great rejections due to the simple reason that each product line has its own good teams, with vast experience and best practices that suit their needs. The change towards commonality is difficult, and at the beginning also time consuming. However, when conducting the proper evangelism, the teams agree that they can benefit each other by a quick understanding of their own pains, receiving (and providing) external assistance, and generating a brotherhood of cross company architects and designers. The usual question of "what's in it for me?" dissipates, and the community uses the common notation as their agreed language, across different products and technologies.

Applying the common standard into the company involves the following meta-architecture activities:

4.1. Unique UML Profile

Proprietary and generic customized UML profiles are constructed according to the company's needs of architectural views. Meaning, definitions of new UML stereotypes are created. Such types can be message calls for SOA, web, notification and publish-subscription, WAN support constrains and so on.

Figure 2 displays partial elements from the Company Profile such as Virtual Layer (enables packaging of conceptual elements and services), Business Service (as defined by s specific domain standard, in Mercury ecosystem the standard is ITIL), a Service (a collection of external activated interfaces such as UI, API, SOA, WS and more), Exposes association, Conceptual Entity (extracted from the domain of requirements modeling), and more.

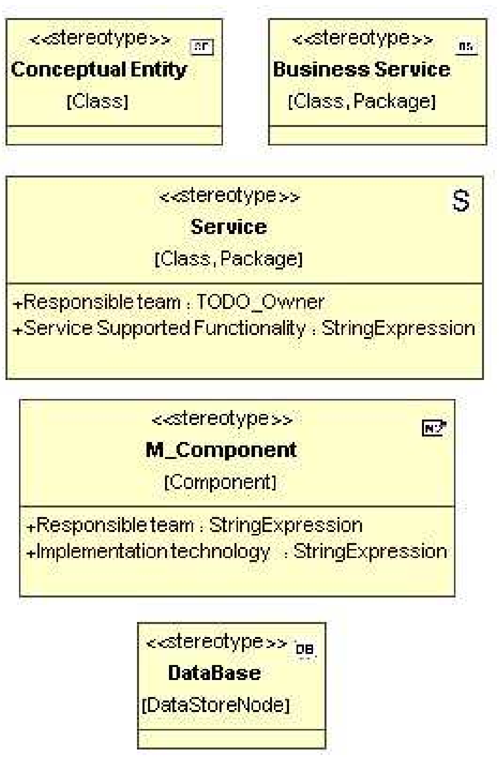

Fig. 2. : Partial Mercury UML Profile which defines new architectural required information and notation, specifically designed for the most abstract layers of the company Architecture

4.2. Architectural Views Templates

Mercury architecture methodology is supported by modified UML profile and company defined architectural diagrams. All of these diagrams are implemented within the Magic Draw UML case tool. These views are decomposition (for functional dependency), layers (for build process and version control), usage (for impact and change analysis), deployment (for configuration management), and concurrency (for load balancing, resource utilization and deadlocks preventions).

We initially construct the products' architecture separately, using the following diagrams according to the logical, conceptual and physical perspectives of the meta-architecture. Implementation of these templates by the designers merely involves the selection of the requested diagram, the same way regular UML diagrams are constructed (see figure 3). The designer selects the appropriate symbols, notations and connection elements from a small set of icons, thus implements quickly the relative perspective needed to be produced.

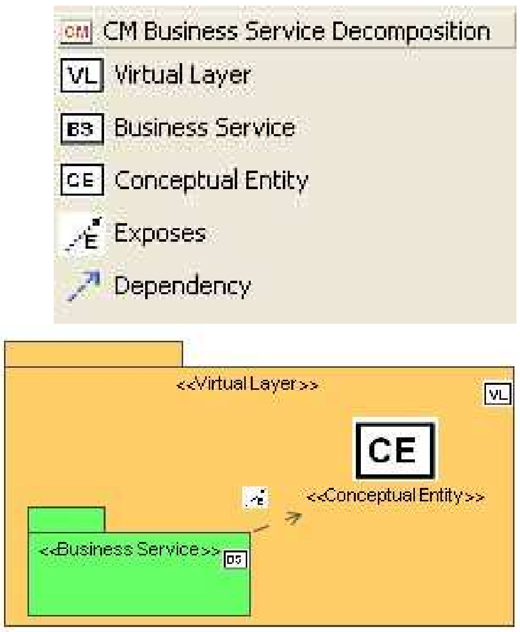

Fig. 3. : Conceptual Modeling Business Service Decomposition Diagram and its relevant modified UML symbols as implemented at Mercury

By modeling products' architecture using these diagrams, we can later on generate specific questions that assist us in our architectural evolution questions such as:
Which products exhibit similar business service? Which common business services does a single product exhibit? What are the common infrastructure services that all products consume?

In order to provide such cross products concerns we use the information inserted within our company cross product model, and extract from it targeted diagrams.

These diagrams serve us in focusing on a specific topic within a review sessions well as for presentation of the different products concerns.

Using the question-driven diagrams, based on the common architectural model, enables us to compare the products and find similarities, duplication and possible common links among them. These observations enable us to define the architectural roadmap of each product separately, as well as common agenda among the products, thus defining the next architectural evolution.

Among the diagrams we are using are:

- Conceptual Mercury Level: Business Service Decomposition; Business Service Usage
- Conceptual Architecture: Service Offering Decomposition; Service Usage Dependency; Conceptual Domain Model (Mercury or Product Level).
- Logical Architecture (System and Functional): BPN Service Activation, Components & Interfaces Usage

4.3. On-the-Job-Training

This topic is probably the most crucial one. In order to maximize the full usage and understanding of what is important and what is not, while preventing the CASE tool from becoming "Shelfware" (Software on the shelf), central architects need to assist the teams in creating their own architectural views while using the automatic CASE tools. This way, the designers are not wasting valuable time, the product development is not at hold, and the knowledge is transferred from one architect to another.

4.4. Implementation Experience

The company conducts regular reviews, which are targeted to a specific concern that the cross architecture team needs to addresses. In these reviews, the product architects are requested to answers specific questions, such as the ones elaborated in section 4.2. Each architect is modeling his or her own product separate model using MagicDraw 11 CASE tool, by using the diagrams templates. According to the discussion generated during the review, the central architecture team clarifies and corrects the information within the model. Namely: Level 0 is defined as cross Mercury architectural concerns (services and components) and Level 1 is defined as the inner product architectural concerns.

The outer perspectives of the product model (Level 0), and its dependency on other products is documented and maintained by the central team, by means of another dedicated consolidated model.

This model holds:

- Level 0 service and components of the products.
- Cross products dependency and usage.

- Mapping of products Services into the Standard Business Model (in Mercury it is the ITIL).
- Mapping of Services and Business Services into Standard Conceptual Model (ITIL conceptual entities).

Based on this constantly updated model, the team and the architectural forum can conduct ongoing analysis of cross concerns, define a common language that serves as the integration language of the products, define technological implementation, and more.

The model serves management debates, architectural reviews and designers, as the estimated state of all products (since the product is constantly changing). It holds the three perspectives of architectural concerns relevant to customer, system and deployment, and maintains constant focus of all teams.

5. Mercury Architecture and Design Lifecycle

This final pillar of Meta-Architecture agenda constitutes the mechanics of project management, and provides company visibility into the products' lines. The impact of assistance during design by expert teams outside the development task force as well as defect detection in the proposed design during reviews, need to be correlated with other teams' daily activities. This task is the most difficult one since it involves resource sharing in a matrix management manner vs. an ad-hock dynamic handling. However, insisting on these cross product peer review generates a common agenda, outer product visibility, and exposure of the products' architects. It is facilitated by a driving force known as the "architects' forum": a regular meetings forum in which the architects present the pains and gains of their architectural plans, get credit for the tasks conducted, and raise common issues and "bad smells" that call for cross management intervention.

When familiarized with each other's needs, the "scratch my back, and I will scratch yours" phenomena is decreasing, and the "all for one and one for all" Musketeers theme becomes the main voice.

The facilitation of an architectural and design process revolves around a customizable flow [5], periodically configured to match the different new features and changes to be inserted at the next release cycle. It is controlled by the Architectural Review Board, which slices the tasks, defines the design teams, defines the type of reviewers needed for these tasks, and delivers theses definitions to the development managers.

Consequently, each iteration cycle is controlled and changed according to the actual outcomes of the reviews and the relevant recommendations.

6. Conclusion

This practitioner paper demonstrates how Mercury, a large R&D organization with a large diversity of products, regards the Architectural Centric Evolution concepts as an ideology, as well as its implementation strategy and tactics.

In order to assist the products' architects in conducting their tasks, Mercury formed the Central Architecture team, and provided them with the professional authority to lead major changes towards unifications, standardization, best methodological practices, and mutual assistance. The company crucial agenda is backed by practical implementations lead by the Meta-Architecture activities: the Mercury Modeling community, the Mercury Modeling Standard, and the Mercury Architecture and the Design Lifecycle. The combination of these responsibilities into a single entity enables us to receive information from all product lines and thus consolidate, adapt, invent, and later on broadcast back, the information as needed by the architects. Moreover, it enables us to adapt to new technologies and methodologies since we consolidate the entire aggregated knowledge of the architects into a single thinking virtual entity with residual mental backups.

Our major tool for enabling decision making regarding the next architectural centric evolution is the product reviews presented and consolidated into a single company model. The Company questions-driven architectural-views generate a focus during these reviews. Accordingly, the team conducts analysis of cross product dependency at the highest abstraction levels and defines the next release roadmap.

Mercury's experience in implementing centric architectural evolution has proved to be efficient. The implementation and outcome of the discussions based on the information extracted from the model, generated a common understanding of the company needs and roadmaps. The investigation of finding the meta-architectural architectural-views, and elaborating them on a single model, generated a true understanding of the architectural required changes. According to these clarifications, the team and management define the goals and relevant agenda of all products, via a single architecture perspective.

Working with Meta-Architecture Centric approach, the best practices conducted within the product lines are leveraged, and the total outcome of the individual products contributions becomes much more than the sum of its parts.

References

1. Brown W. J., Malveau R C., McCormick H W., Mowbray T. J., AntiPatterns - Refactoring Software, Architectures, and Projects in Crisis. John Wiley & Sons, Inc., Reading, ISBN: 0471197130, (1998)
2. Cinne'ide, M. & Nixon, P., Composite Refactorings for Java Programs. Proceedings of the Workshop on Formal Techniques for Java Programs, European Conference on Object-Oriented Programming, Univ. College Dublin, (2000).
3. Fowler M. Refactoring: Improving the Design of Existing Code. Addison-Wesley, Reading 1999.

4. Hadar E. and Hadar I., "The Composition Refactoring Triangle (CRT) Practical Toolkit: From Spaghetti to Lasagna", accepted to international conference OOPSLA Portland, Oregon, USA, October 2006
5. Poppendieck M., Poppendieck T. Lean Software Development: An Agile Toolkit. Addison Wesley, May 08, Reading ISBN: 0321150783, (2003)
6. Zdun U. Avgeriou P. Architecture-Centric Evolution. Workshop on Architecture-Centric Evolution (ACE 2005), Hosted at the 19th European Conference on Object Oriented Programming, Glasgow, UK, (ECOOP 2005), 25-29th July 2005.