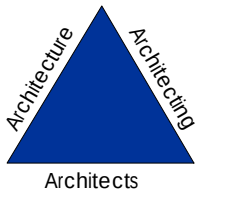


The Visual Architecting Process

The Visual Architecting Process (VAP) incorporates the essential steps involved in creating a good architecture and gaining organizational support and compliance. It uses group graphics and visual models as key mechanisms to gather input, facilitate team collaboration, and enhance communication with architecture stakeholders. This paper outlines the process.

by Ruth Malan and Dana Bredemeyer
Bredemeyer Consulting
ruth_malan@bredemeyer.com
dana@bredemeyer.com

The Visual Architecting Process

The architecting process incorporates a technical process and an organizational process. The technical process includes steps and heuristics for creating a good architecture. However, a technically good architecture is not sufficient to ensure the successful use of the architecture, and the organizational process is oriented toward ensuring support for, and adoption of, the architecture.

The Technical Process

Overview

The focal deliverable of the architecting process is the architecture document (set), motivating and describing the structure of the system through various views. However, though system structuring is at the heart of the architecting process, it is just one of several activities critical to the creation of a good architecture. Architectural requirements are needed to focus the structuring activities. Different architectural approaches tend to yield differing degrees of fit to various system requirements, and evaluating alternatives or performing architectural trade-off analyses should be an integral part of the structuring phase. Lastly, a validation phase provides early indicators of, and hence an opportunity to resolve, problems with the architecture.

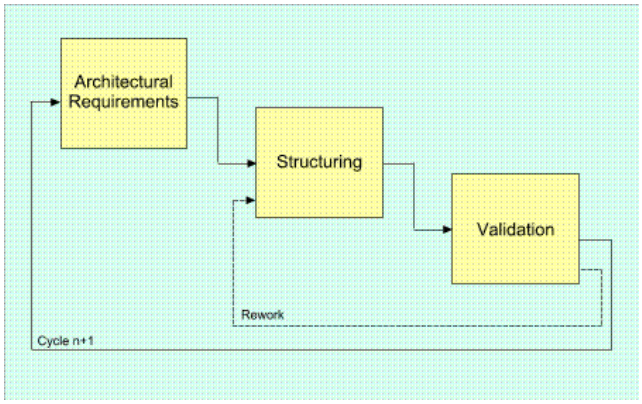


Figure 1. The Technical Architecting Process

Architectural Requirements

Architectural requirements are a subset of the system requirements, determined by architectural relevance. The business objectives for the system, and the architecture in particular, are important to ensure that the architecture is aligned with the business agenda. The system context helps determine what is in scope and what is out of scope, what the system interface is, and what factors impinge on the architecture. The system value proposition helps establish how the system will fit the users' agenda and top-level, high-priority goals. These goals are translated into a set of use cases, which are used to document functional requirements (Malan and Bredemeyer, 1999). The system structure fails if it does not support the services or functionality that users value, or if the qualities associated with this functionality inhibit user performance or are otherwise unsatisfactory. System qualities that have architectural significance (e.g., performance and security, but not usability at the user interface level) are therefore also important in directing architectural choices during structuring (Malan and Bredemeyer, 2001).

Of course, requirements may already have been collected by product teams. In that case, the architecture team needs to review those requirements for architectural relevance and completeness (especially with respect to non-functional requirements), and be concerned with requirements for future products that the architecture will need to support.

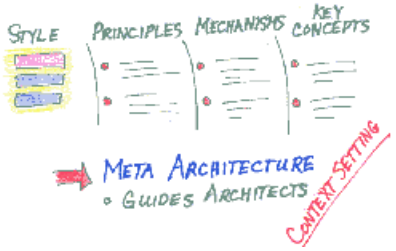
Lastly, for the architecture of a product line or family, architectural requirements that are unique to each product, and those that are common across the product set, need to be distinguished so that the structure can be designed to support both the commonality and the uniqueness in each product.

Architecture Specification

The architecture is created and documented in the architecture specification phase. This is decomposed into sub-phases, along the lines of our model of software architecture:

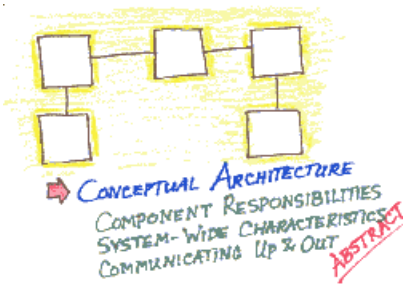
Meta-Architecture

First, the architectural vision is formulated, to act as a beacon guiding decisions during the rest of system structuring. It is a good practice to explicitly allocate time for research—or scavenging for ideas—in documented architectural styles, patterns, dominant designs and reference architectures, other architectures your organization, competitors, partners, or suppliers have created or you find documented in the literature, etc. Based on this study, and your and the team's past experience, the meta-architecture is formulated. This includes the architectural style, concepts, mechanisms and principles that will guide the architecture team during the next steps of structuring.



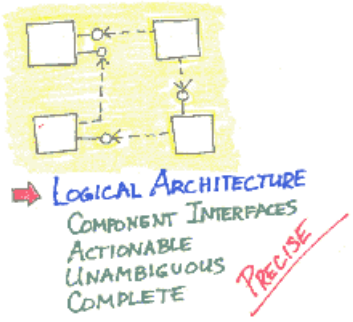
Conceptual Architecture

The system is decomposed into components and the responsibilities of each component, and interconnections between components are identified. The intent of the conceptual architecture is to direct attention at an appropriate decomposition of the system without delving into the details of interface specification and type information. Moreover, it provides a useful vehicle for communicating the architecture to non-technical audiences, such as management, marketing, and many users.



Logical Architecture

The conceptual architecture forms the starting point for the logical architecture, and is likely modified as well as refined during the course of the creation of the logical architecture. Modeling the dynamic behavior of the system (at the architectural—or component—level) is a useful way to think through and refine the responsibilities and interfaces of the components. Component specifications make the architecture concrete. These should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.



Execution Architecture

An execution architecture is created for distributed or concurrent systems. It is formed by mapping the components onto the processes of the physical system. Different possible configurations are evaluated against requirements such as performance and scaling.

Architecture Trade-off Analysis

At each step in architecture specification, it is worthwhile challenging the team's creativity to expand the solution set under consideration, and then evaluating the different architectural alternatives against the pri-

oritized architectural requirements. This is known as architecture trade-off analysis (Barbacci et. al., 1998), and it recognizes that different approaches yield differing degrees of fit to the requirements. Selection of the best solution generally involves some compromise, but it is best to make this explicit.Back to Top

Architecture Validation

During architecture specification, the architects obviously make their best effort to meet the requirements on the architecture. The architecture validation phase involves additional people from outside the architecting team to help provide an objective assessment of the architecture. In addition to enhancing confidence that the architecture will meet the demands placed on it, including the right participants in this phase can help create buy-in to the architecture. Architecture assessment involves "thought experiments", modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience. Another important part of validation is the development of prototypes or proofs-of-concept. Taking a skeletal version of the architecture all the way through to implementation, for example, is a really good way to prove out aspects of the architecture.

Iterations

Though described sequentially above, the architecting process is best conducted iteratively, with multiple cycles through requirements, structuring and validation. One approach is to have at least one cycle devoted to each of Meta, Conceptual, Logical, and Execution architecture phases, and cycles for developing the architectural guidelines and any other materials to help in deploying the architecture (such as tutorials). At each cycle, just enough requirements are collected to proceed with the next structuring step, and validation concentrates on the architecture in its current phase of maturity and depth.

Moreover, a number of architecture teams that we have worked with, have stopped at different points, leaving more detailed architecting to the product and component teams. At one end of the spectrum, a very small team of architects created the meta-architecture, and each of the product teams created their own architectures within the guidelines and constraints of the meta-architecture. Other architecture teams have created the meta- and conceptual architectures, and a broader team of component owners developed the logical architecture. At the other end of the spectrum, the architecture team developed the entire architecture, all the way to its detailed logical architecture specification. This approach yields the most control over the architecture specification, but is typically fraught with organizational issues (e.g., the "NIH syndrome") that slow or even completely inhibit the use of the architecture.

The Organizational Process

Overview

Architecture projects are susceptible to three major organizational sources of failure—the project is under-resourced or cancelled prematurely by an uncommitted management; it is stalled with endless infighting or a lack of leadership; or the architecture is ignored or resisted by product developers. The organizational process helps address these pitfalls. Two phases—namely Init/Commit and Deployment—bookend the technical process.

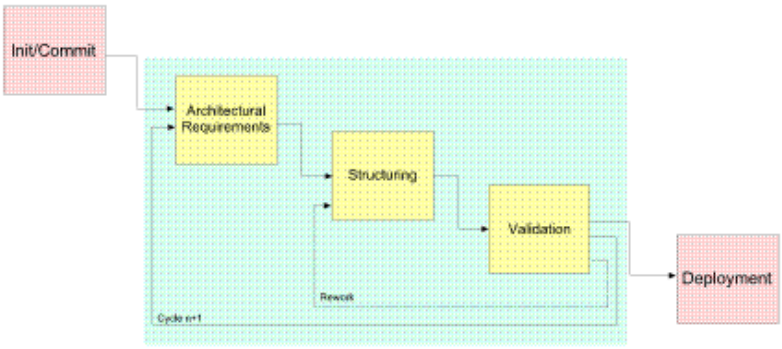


Figure 2. The Architecting Process including the Organizational Process

However, the principal activities in these phases, namely championing the architecture and leading/teaming in Init/Commit, and consulting in Deployment, also overlap with the technical process activities (Figure 2).

Init/Commit

The Init/Commit phase focuses on initiating the architecture project on a sound footing, and gaining strong commitment from upper management. The creation of the architecture vision is central both to aligning the architecture team and gaining management sponsorship. A communication plan is also helpful in sensitizing the team to the need for frequent communication with others in the organization. A heads-down, hidden skunkworks architecture project may make quick progress—as long as it is well-led and its members act as a team. However, not listening to the needs of the management, developers, marketing, manufacturing and user communities and not paying attention to gaining and sustaining sponsorship in the management and technical leadership of the organization, or buy-in from the developer community, will lead to failure. The communication plan places attention on balancing the need for communication and isolation, as well as planning what to communicate when, and to whom.

Deployment

The Deployment phase follows the technical process, and addresses the needs of the developers who are meant to use the architecture to design and implement products. These range from understanding the architecture and its rationale, to responding to the need for changes to the architecture. This entails consulting, and perhaps tutorials and demos, as well as the architects' involvement in design reviews.

Championing

It is important that at least the senior architect and the architecture project manager (if there is one) champion the architecture and gain the support of all levels of management affected by the architecture. Championing the architecture starts early, and continues throughout the life of the architecture, though attention to championing tapers off as the architecture comes to be embraced by the management and developer communities.

Leading/Teaming

For the architecture team to be successful, there must be a leader and the team members must collaborate to bring their creativity and experience to bear on creating an architecture that will best serve the organization. This would seem so obvious as to not warrant being said, but unfortunately this is easier said than done. Explicit attention to developing the designated lead architect’s leadership skills, in the same way one would attend to developing these skills in management, is a worthy investment. Likewise, investing in activities aimed at developing the team as a team also has great payoff in the team’s efficacy.

Communicating and Consulting

Consulting with and assisting the developer community in their use of the architecture is important in facilitating its successful adoption and appropriate use. These activities are most intense during deployment. However, earlier communication and consulting helps create buy-in the developer community through participation and understanding. This allows the architecture team to understand the developers’ needs and the developers to understand the architecture (and its rationale) as it evolves through the cycles of the technical process.

Conclusion

We have worked with and studied dozens of architecting projects, and distilled what we believe to be the best practices and pitfalls that would help architects successfully create and deploy their architectures. This experience guided our creation of the Visual Architecting Process. Though our architecting process lays out the activities and guidelines that we have derived from real-world experience, no project that we studied followed exactly this process. Also, every project that we have consulted with or coached, has adapted the process. This has been true of other software development methods, such as SA/SD, OMT and Fusion (Malan, Coleman and Letsinger, 1995). It would appear that a method is not fully embraced by a project team until they have adapted it to their particular project needs. In this regard, methods are somewhat like architectures! Actually, we strongly encourage you to tailor "just enough process" to meet your project goals and current context. Keep your audience in mind, and orient what you do so that it *can* and *will* be used.

References

Barbacci, M. R., S. J. Carriere, P. H. Feiler, R. Kazman, M. H. Klein, H. F. Lipson, T. A. Longstaff, and C. B. Weinstock, "Steps in an Architecture Trade-off Analysis Method: Quality Attribute Models and Analysis", http://www.sei.cmu.edu/publications/documents/97.reports/97tr029/97tr029title.htm See also the *SEI Architecture Trade-off Analysis Initiative* website, http://www.sei.cmu.edu/ata/ata_init.html

Malan, R. A., R. Letsinger and D. Coleman. "Lessons Learned from Leading-Edge Object Technology Projects in Hewlett-Packard", *Proceedings of OOPLSA'95*, 1995.

Malan, R. A., and D. Bredemeyer, "Defining Non-Functional Requirements", published on the *Resources for Software Architects* web site at http://www.bredemeyer.com/pdf_files/NonFunctReq.PDF, 39kb, August 2001.

Malan, R. A., and D. Bredemeyer, "Functional Requirements and Use Cases", published on the *Resources for Software Architects* web site at http://www.bredemeyer.com/pdf_files/functreq.pdf, 39kb, June 1999.

Ogush, M., D. Coleman, and D. Beringer, "A Template for Documenting Software Architectures", published on the HP architecture web site which has been discontinued, March 2000.

Youngs, R., D. Redmond-Pyle, P. Spaas, and E. Kahan, "A Standard for Architecture Description", *IBM Systems Journal*, Vol 38 No 1. http://www.research.ibm.com/journal/sj/381/youngs.html

Resources for Software Architects

The *Resources for Software Architects* web site (http://www.bredemeyer.com) organizes a variety of resources that will help you in your role as an architect or architecture project manager. A number of Bredemeyer Consulting's *Action Guides* and *Architecting Process Action Guides* are on the papers and Downloads page (http://www.bredemeyer.com/papers.htm). You will also find books and papers listed under "How: The Architecting Process" in our Bibliography, and on the Architecture Books List (http://www.bredemeyer.com/Books/recommendedBooks.htm). You may also be interested in our *Software Architecture Workshop*, as well as our *Architecture Vision Workshop*. For more information, please see http://www.bredemeyer.com/training.htm.

Glossary

Architecture. The term software architecture is used both to refer to the high-level structure of software systems and the specialist discipline or field distinct from that of software engineering. The architecture of a software system identifies a set of components that collaborate to achieve the system goals. The architecture specifies the "externally visible" properties of these components—i.e., those assumptions other components can make of a component, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on (Bass et. al., 1998). It also specifies the relationships among the components and how they interact. It is best expressed in terms of a set of views.

Conceptual Architecture. The intent of the conceptual architecture is to direct attention at an appropriate decomposition of the system without delving into the details of interface specification and type information. Moreover, it provides a useful vehicle for communicating the architecture to non-technical audiences, such as management, marketing, and many users. The conceptual architecture identifies the system components, the responsibilities of each component, and interconnections between components. The structural choices are driven by the system qualities, and the rationale section articulates and documents this connection between the architectural requirements and the structures (components and connectors or communication/co-ordination mechanisms) of the architecture.

Functional Requirements. Functional requirements capture the intended behavior of the system—or what the system will do. This behavior may be expressed as services, tasks or functions the system is required to perform. Use cases have quickly become a widespread practice for capturing functional requirements.

Logical Architecture. The logical architecture is the detailed architecture specification, precisely defining the component interfaces and connection mechanisms and protocols. It is used by the component designers and developers.

Meta-architecture. The meta-architecture is a set of high-level decisions that will strongly influence the structure of the system, but is not itself the structure of the system. The meta-architecture, through style, patterns of composition or interaction, principles, and philosophy, rules certain structural choices out, and guides selection decisions and trade-offs among others. By choosing communication or co-ordination mechanisms that are repeatedly applied across the architecture, a consistent approach is ensured and this simplifies the architecture. (See http://www.bredemeyer.com/howto.htm and http://www.bredemeyer.com/whatis.htm.)

Non-functional Requirements. Non-functional requirements or system qualities, capture required properties of the system, such as performance, security, maintainability, etc.—in other words, how well some behavioral or structural aspect of the system should be accomplished.

Product Line. Product lines consist of basically similar products with different cost/feature variations per product.

Product Family. Product families include a number of product lines targeted at somewhat different markets or usage situations. What makes the product lines part of a family, are some common elements of functionality and identity.

Qualities. System qualities, or non-functional requirements, capture required properties of the system, such as performance, security, maintainability, etc.--in other words, how well some behavioral or structural aspect of the system should be accomplished.

Use Case. A use case defines a goal-oriented set of interactions between external actors and the system under consideration. That is, use cases capture who (actors) does what (interactions) with the system, for what purpose (goal). A complete set of use cases specifies all the different ways to use the system, and thus defines all behavior required of the system—without dealing with the internal structure of the system.

About Bredemeyer Consulting

Bredemeyer Consulting provides a range of consulting and training services focused on Enterprise, System and Software Architecture. We provide training and mentoring for architects, and typically work with architecture teams, helping to accelerate their creation or migration of an architecture. We also work with strategic management, providing consulting focused on developing architectural strategy and organizational competency in architecture.

BREDEMEYER CONSULTING
Bloomington, IN 47401
Tel: 1-812-335-1653
http://www.bredemeyer.com