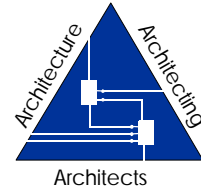


ARCHITECTURE RESOURCES

For Enterprise Advantage

<http://www.bredemeyer.com>



BREDEMEYER CONSULTING, Tel: (812) 335-1653

The *Visual Architecting Process*[™]

by Ruth Malan and Dana Bredemeyer

Bredemeyer Consulting

ruth_malan@bredemeyer.com

dana@bredemeyer.com

The *Visual Architecting Process*[™] (VAP) integrates what we have learned from great architects, and failed architectures, to help you create a software architecture that is:

- *good*—it is technically sound and clearly represented
- *right*—it meets the needs and objectives of key stakeholders, and
- *successful*—it is actually used in developing systems that deliver strategic advantage.

The *Visual Architecting Process*[™] incorporates the essential steps involved in creating a good, right 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.

From "Chapter 2. The Visual Architecting Process: Good, Right and Successful". The book is titled *Software Architecture Action Guide*, by Ruth Malan and Dana Bredemeyer.

Introduction

Everywhere software development is discussed, we hear the same litany of concerns: we face unpredictability in development schedules and system behavior, and quality issues with hard to find defects, hard to fix problems, and fixes that introduce new problems. Our systems are increasingly difficult to change, making it harder to respond to market shifts. Software development costs more and takes longer. Reuse opportunities are lost because it is hard to isolate chunks to reuse, or the chunks are highly tuned to a specific product. There are morale problems and it is hard to keep good people, and hard to energize teams for the next brutal round trying to meet any kind of schedule. All this has the result that we are losing ground in the market, our competition can do more with less, and we are not responsive to our customers. If it wasn't so painful to all of us embroiled in the problems of software development, this constant repetition of our woes would get quite tiresome.

The whole situation is exacerbated because, in spite of this list, we are generally quite successful. Our market tolerates our schedule slips and grudgingly bears with us through our quality issues, and our organization goes through round after round of bloating growth coupled with displays of developer heroism. But with each success, our pain gets worse, until suddenly we find we are not successful, and we fail imploratively!

Over the past decade or more, the same incantation of woes has motivated SA/SD, OO, OOA/D, UML, XP, and, yes, even architecture. And various combinations of the same solutions have been posited: modeling, encapsulation and information hiding, small teams, iterative development.

Certainly, these are all part of the answer. But the solution has to be to start with architecture, and to *follow it through!* This is important for a small project, and indispensable for a large project. For projects of any complexity (due to the number of people involved, difficulty of the problem, broad scope, etc.), architecture presents the only way to address significant issues. But it is insufficient to create a “box and line” drawing showing the system decomposition, although this is a step along the way to partitioning the problem and making it comprehensible and manageable. It is even insufficient to create well-crafted and documented interfaces, although this is a step along the way to making the architecture precise and actionable. A great architecture is simply a bundle of paper until it is implemented. Along the way, all manner of intentional and unintentional “accommodations” to the architecture are likely to compromise its structural integrity and render those elegant mechanisms unrecognizable—unless the organization accepts and understands the architecture and the costs of subverting it.

And so we come to the Visual Architecting Process. It will not make you successful—only you can do that. But it will help you know what to do and how to do it.

The *Visual Architecting Process*[™]

In a companion paper titled “Software Architecture: Central Concerns, Key Decisions” (Malan and Brede-meyer, 2002), we consider what concerns software architecture addresses, and how one expresses software architecture. But all the documentation and presentations in the world will not suffice, unless the software architecture is:

- *good*—it is technically sound and clearly represented
- *right*—it meets the needs and objectives of key stakeholders, and
- *successful*—it is actually used in developing systems that deliver strategic advantage.

The *Visual Architecting Process*[™] (VAP) covers the techniques, including architecture modeling and architecture trade-off analysis, used in creating a technically sound architecture. It covers architectural requirements and prioritization to create the right architecture, together with architecture validation to ensure that the architects and key stakeholders agree that it is indeed the right architecture. And it covers

the organizational process steps that help ensure that the architecture is embraced and used informedly so that, ultimately, the architecture is successful.

Figure 1 provides a high-level overview of VAP. For a more full depiction, see the VAP poster on our web site (http://www.bredemeyer.com/pdf_files/VisualArchitectingProcess_Core.PDF).

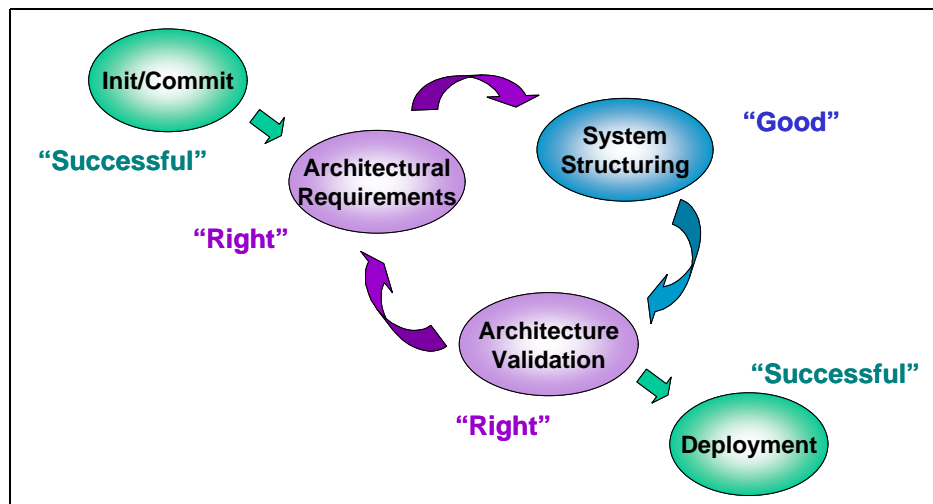


Figure 1. The Visual Architecting Process™

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.

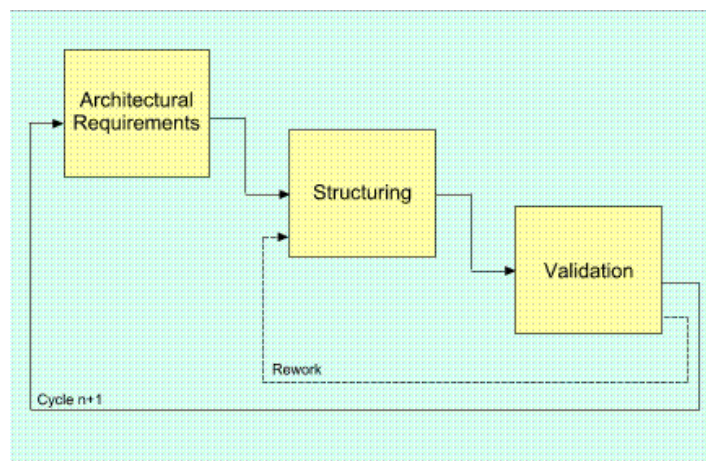


Figure 2. The Technical Architecting Process

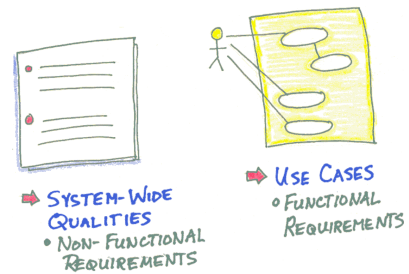
Architectural Requirements

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 neglect other stakeholder goals. 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, August 2001).

Of course, requirements may already have been collected by product teams or business analysts. 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.



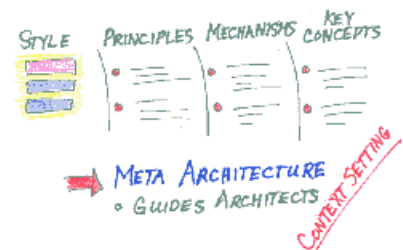
Architecture Specification

The architecture is created and documented in the architecture specification phase. This is decomposed into sub-phases, addressing sets of decisions along the lines of our software architecture decision model (Malan and Bredemeyer, 2002).

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.

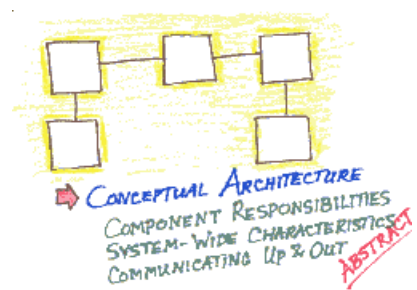
First, the architectural vision is formulated, to act as a beacon guiding decisions during the rest of system structuring. It is a good practice to explicitly allocate time for research—or scavenging for ideas—in documented architectural styles, patterns, dominant designs and reference architectures, other architectures that 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 intent of the conceptual architecture is to direct attention at an appropriate decomposition of the system before 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.

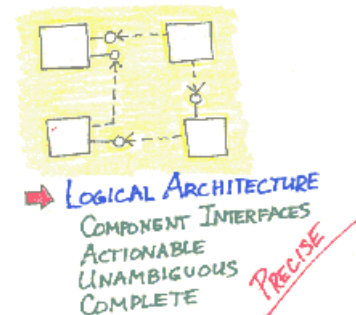
During the conceptual architecture phase, the system components are identified, along with the responsibilities of each component, and interconnections between components. These structural choices are driven by the system qualities, and are document in the rationale section which articulates this connection between the architectural requirements and the structures (components and connectors or communication/co-ordination mechanisms) of the architecture.



Logical Architecture

The logical architecture is the detailed architecture specification, precisely defining the component interfaces and connection mechanisms and protocols. It also details out other mechanisms designed to address cross-cutting concerns. It is used by the component designers and developers.

The conceptual architecture forms the starting point for the logical architecture, and it 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. The process view shows the mapping of components onto the processes of the physical system, with attention being focused on such concerns as throughput and scalability. The deployment view shows the mapping of (physical) components in the executing system onto the nodes 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. The different architectural alternatives are then evaluated against the prioritized 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.

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 (such as tutorials) to help in deploying the architecture (Figure 3). 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 .

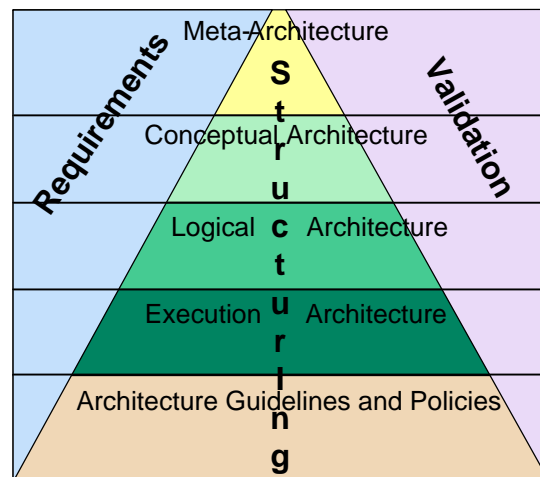


Figure 3. Schematic overlaying the Visual Architecting Process upon the Architecture Decision Model

We also do not recommend turning the process into a “waterfall” by sticking rigidly to the bands in the Architecture Decision Framework (Malan and Bredemeyer, May 2002), completing all of Meta-Architecture before moving on to Conceptual Architecture, for example. Rather, use the Decision Model as a guide, and as a placeholder for completed work (system abstractions and concepts go in the Conceptual Architecture document set, interface details go in Logical architecture, etc.). But by all means move forward quickly to investigate in more detail aspects of the architecture that are high risk. As you work decisions in another area (perhaps you are exploring the design of a key architectural mechanism, and thinking through the behavior of components involved) you will likely rethink or have new insights into areas already “dealt with”. It only makes sense to hold the whole architecture somewhat in suspension as you move through the process, being willing to backtrack and make changes, then work forward again.

However, what you present to external audiences is a systematic process that moves in an organized way through the layers of decisions, from strategy to conceptual views to more precise and actionable architecture specifications. You can still have program checkpoints (with formal reviews and sign-off to mark completion of Meta, Conceptual, Logical, Execution, Guidelines) to project this organized progress,

although behind the scenes you have already scouted out the architectural territory that lies ahead of the official checkpoint.

Another permutation to the process has to do with how much of this decision set the architecture team completes in a small focused team, versus expanding the team or even handing off the remaining architectural decisions to the development teams. 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 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—

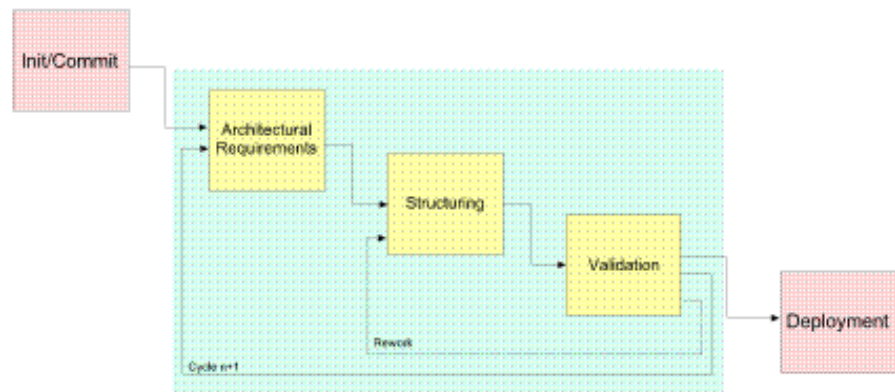


Figure 4. The Architecting Process including the Organizational Process

bookend the technical 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 4).

Init/Commit

The Initiate and gain Commitment (or 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.

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 may taper 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 (Malan and Bredemeyer, March 2001) 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 far less evident in practice than one would suppose. In many organizational cultures, there is collusion among the technical people not to allow a leader to emerge. When we point out the lack of leadership, we are told “this is a consensus culture,” but really it is a culture that has formed an effective way to allow no-one to lead by insisting that all must agree to every decision. The reality is there will be logjams, that is the nature of this game. Someone has to lead the team through tough decisions. Others have to follow. We’ve said it before, and we’ll say it again: goodwill is the real silver bullet.

We believe in the value of collaboration and consensus, and we are pragmatists. Sometimes the consensus process is too slow, or inherently the issues are too divisive. In an architecture team full of talented people with strong leadership qualities and high emotional investment in the goodness and success of the architecture, there nonetheless needs to be one person who is appointed to be the leader, and accepted as the leader among the team and the rest of the organization. The architecture team needs to set the example of good following, for how can they expect the development community to follow the architecture if they themselves display poor following behavior?

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.

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 among 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.

Rather than protest that it is the responsibility of developers to read the architecture documents, architects need to play a proactive role. Workshops designed around exercises applying the architecture are great at getting participants’ minds and hands engaged. It also takes ongoing consulting and constant educating and influencing through the construction phase of the project to ensure the architecture is understood and adhered to, and to take corrective action when ambiguities or misunderstandings surface. The architect’s role at this point is guide and coach, generally setting direction but drawing a firm line when needed.

Further, the architect needs to be attentive to any emerging decisions that require architecture intervention or assistance. The architect also serves as a resource to the developers—a sounding board, someone to assist with problem-solving, a mentor. The more effective and credible the architect, the more likely it will be that developers seek the participation of the architect when they encounter challenges that might be architectural (broad scope, high impact, or affecting system integrity). This provides the architect with much-needed opportunities to stay current with the challenges emerging in the domain, while providing developers the benefit of the architect’s talent and experience and deep insight into the architecture.

Principles to Guide the Architecting Process

How far to go in architecting a system requires judgment and experience, but we have established a set of principles to help you make the call between more architectural detail and hence more constraints being placed on the development community, versus less detail and hence less control over the system formulation and ultimate achievement of the architecture vision, objectives and key properties.

Minimalist Architecture Principle

Essentially, the *Minimalist Architecture Principle* says “if a decision can reasonably be made by someone with a more narrow scope of responsibility, defer the decision to that person or group” (Malan and Brede-meyer, 2002). This means that architects only make decisions that *require* the overall perspective and insight of the architect, are critical to the achievement of strategic business objectives at the scope of the architecture, and if delegated, would likely be made in a way that compromises the achievement of those objectives. If a decision does not meet these criteria, then the architect should not make it.

Decisions With Teeth Principle

Another way of pruning the architecture decision set is to apply the *Decisions with Teeth Principle*. Too often, architects and their deployment communities treat architecture decisions as statements of “general good.” These decisions are treated like guidelines or suggestions, which other architects, designers or implementers choose whether or not to follow. Such decisions have no teeth. The reason to avoid making decisions that are likely to be dismissed is simple: put bluntly, it is a waste of time for the architect to make, and for others to think about and then ignore, such decisions. And it is damaging to breed a culture that supports an architecture free-for-all. You cannot achieve your architecture goals without an effective architecture decision set.

You can give a decision teeth if you are passionate enough about the decision to be its champion and its watchdog. But each decision that relies on you to give it teeth, diminishes your overall effectiveness—you just get spread too thin to keep up. The governance process is intended to shift responsibility for providing all of the teeth from the architect to others who share the burden. Now, decisions that have teeth are those that are both enforceable and enforced. The governance process allows for discovery of breaches and determines consequences, rather than granting exceptions at every request. This does mean that the decisions must be well-formed. They have to be unambiguous and have a clear scope of applicability.

This may highlight a need to improve your governance process, but even with a strong governance process in place, objections raised in the name of customer advocacy have a powerful shield. That is, arguments in favor of the “general good” are susceptible to counter-arguments made in the name of a particular customer (or market segment) or immediate concern.

Connect-the-Dots Principle

Now here’s the rub: each decision in a truly minimalist architecture is there because it could not be made by someone with local scope of responsibility—if they did, they would compromise the overall goals of the architecture. In other words, everyone else either does not have the perspective and knowledge to make

the decision, or they would make a decision that maximizes their local interests at the expense of the architecture goals. By their nature, decisions in a minimalist architecture are going to have detractors who, from their local perspective, see the decisions as sub-optimal. In short, the decisions that are worth making part of the architecture are exactly the ones that will be controversial in some quarters.

This presents a conundrum which is resolved by applying another principle, which we call *Connect-the-Dots*. According to this principle, each architecture decision must be rationalized in terms of business goals, system and architecture requirements, or higher-level architecture decisions. You see, the only voice that stands any chance of holding its own against the voice of a customer is the voice of the business. Business strategy represents the voice of the business, and connect-the-dots creates a compelling chain that links business strategy to architecture goals to architecture decisions.

At its best, business strategy is well grounded in the voice of the customer *and* it is grounded in the voice of the business telling the story of competitive differentiation. It takes into account the competitive environment, the value network, internal capabilities and financial goals. Architecture that takes business strategy as its starting point, and shows how each architecture decision is necessary to achieve the business strategy, expresses the voice of the business, and follows the connect-the-dots principle.

Architecture as the Technical Expression of Business Strategy

When the case has been made that the architecture decisions satisfy these three principles, then that set of decisions can be described as the technical expression of the business strategy. When such a decision, clearly driven by the business strategy, is ignored, we need to realize that it is not the architecture that is being brought into question, but the strategy itself. This focuses discussion where it belongs. We need to not get distracted by debate about technology questions when the real issue is clarifying and enforcing what is strategic to the business.

With a minimalist architecture, and connected dots, we can empower the governance process to provide the teeth that will make the architecture stick. Hopefully, though, we can rely more on the bark than the bite, and still more on persuasion, relying on the goodwill of those whose immediate interests are somewhat compromised because the overall benefit of achieving the business strategy makes it all worthwhile.

With a bloated architecture, no matter how well-intentioned, it all comes unraveled. A bulky architecture is too expensive and hard to give teeth to. It is expensive to read and expensive to execute on, not to mention more likely to be flawed. Your governance process will get locked in interminable exception processing allowing no bandwidth to catch important deviations from the architecture nor any shortfall in the architecture itself.

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 your architecture *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, "Architecture Teams", published on the *Resources for Software Architects* web site at http://www.bredemeyer.com/pdf_files/ArchitectureTeams.PDF, 39kb, March 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.
- Malan, R. A. and D. Bredemeyer, "Software Architecture: Central Concerns, Key Decisions", published on the *Resources for Software Architects* web site at http://www.bredemeyer.com/pdf_files/ArchitectureDefinition.pdf, May 2002.
- Malan, R. A. and D. Bredemeyer, "Less is More with Minimalist Architecture", *IT Professional*, IEEE, September 2002.
- 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>

Restrictions on Use

This paper and all other material that is published on the *Resources for Software Architects* web site (<http://www.bredemeyer.com>), may be downloaded and printed for *personal* use. If you wish to quote or paraphrase fragments of our work in another publication or web site, please properly acknowledge us as the source, with appropriate reference to the article or web page used. If you wish to republish any of our work, in any medium, you must get written permission from the lead author or the site editor. Also, any commercial use must be authorized in writing by Bredemeyer Consulting.

A Note About the Forthcoming Book

We are writing a book for software architects that is short and oriented to guiding action. It has two parts, with the first part providing context and a guide to the process. The second part is the full set of Action Guides, one for each discrete technique, model or template that is used in the Visual Architecting Process. For a preview of our Action Guides, please look at examples under Downloads on the *Papers and Downloads* page of our web site at <http://www.bredemeyer.com/papers.htm>.

Please join our mailing list to receive notice of new chapters as they are added to the site. To do so, complete the form on our web site at <http://www.bredemeyer.com/Forms/subscrib.htm>, or click the envelope icon in the sidebar of most pages on our web site.

Table of Contents

Part I: Software Architecture and the Visual Architecting Process

- Chapter 1. Software Architecture: Central Concerns, Key Decisions
- Chapter 2. The Visual Architecting Process: Good, Right and Successful
- Chapter 3. Initiate and Gain Commitment: Getting Started
- Chapter 4. Meta-Architecture: Getting Strategic
- Chapter 5. Conceptual Architecture: Getting the Big Chunks Right
- Chapter 6. Logical Architecture: Getting Precise, Making Actionable
- Chapter 7. Execution Architecture: Getting Physical
- Chapter 8. Architecture Guideline and Policies: Getting Specific
- Chapter 9. Architecture Deployment: Getting Real

Part II: Software Architecture Action Guides

Here are some examples of what we call Action Guides:

- Software Architecture Principles Template (http://www.bredemeyer.com/pdf_files/Principles_Template.PDF, 24kb)
- Stakeholder Profile Action Guide (http://www.bredemeyer.com/pdf_files/Stakeholder_Profile.PDF, 222kb)
- Use Case Template (http://www.bredemeyer.com/pdf_files/UseCase_Template.PDF, 25kb)
- Interface Specification Template (http://www.bredemeyer.com/pdf_files/Interface_Template.PDF, 24kb)

Resources for Architects

Software Architecture Workshop. This class focuses on the Visual Architecting Process (VAP). It is organized around the process. As the workshop progresses, small teams of participants take their project from vision to architecture. This format, punctuating lectures with exercises that build on one another, gives participants the opportunity to learn and practice techniques used in each of the process steps.

Open Enrollment Class:

Indianapolis, IN on **September 26-29, 2005**

London, UK on **June 20-23, 2005**

See http://www.bredemeyer.com/architecture_workshop_overview.htm

Enterprise Architecture Workshop. This class focuses on the Visual Architecting Process for the *Enterprise* (VAP-Enterprise). Following a couple of context-setting modules, the core sections of the course are organized around the process. It follows a workshop format, with lecture modules followed by team exercises to practice techniques and solidify concepts and models. The Visual Architecting Process for the Enterprise starts with Business Strategy, identifies and refines the Business Capabilities Architecture, and uses this to drive the Information (data), Application Solution, and Technology (Infrastructure) Architectures at the enterprise level of scope.

Open Enrollment Class:

Palo Alto, CA on **March 21-24, 2005.**

London, UK on **June 20-23, 2005**

See http://www.bredemeyer.com/Enterprise_Architecture/Enterprise_Architecture_Workshop.htm

Software Architecture Overview Seminar: This course goes over the basics (what, why, how, who, when, and where) and is a good introduction for managers and developers, business analysts and others who need to partner with architects in making architecture successful. This class will build insight into the importance of architecture, as well as the role of architects *and others* in making architecture successful.

See <http://www.bredemeyer.com/Workshops/Descriptions/architectureConceptsWorkshop.htm>

Role of the Architect Workshop. Excellent class for architects—according to those who have taken it, that is. This class helps you identify what skills you need to strengthen, and starts you along the road to building them, while providing options for what to do next to further develop needed skills.

Open Enrollment Class: Indianapolis, IN on **May 24-26, 2005**

See http://www.bredemeyer.com/role_of_architect_workshop_overview.htm

For more information on our training classes, please see <http://www.bredemeyer.com/training.htm>.

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 white papers and *Action Guides* are on the Papers and Downloads page (<http://www.bredemeyer.com/papers.htm>).

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 renovation of an architecture. We also work with strategic management, providing consulting focused on developing architectural strategy and organizational competency in architecture.

We manage the ***Resources for Software Architects*** web site (see <http://www.bredemeyer.com>). This highly acclaimed site organizes a variety of resources that will help you in your role as architect or architecture program manager. A number of Bredemeyer Consulting's *Action Guides*, presentations and white papers are on the Papers and Downloads page (<http://www.bredemeyer.com/papers.htm>). You may also be interested in our *Software Architecture* and *Enterprise Architecture* Workshops, as well as our *Architectural Leadership* class. For more information, please see <http://www.bredemeyer.com/training.htm>.

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