ARCHITECTURE RESOURCES
For Enterprise Advantage
http://www.bredemeyer.com

Architecture Architecting Architects

BREDEMEYER CONSULTING, Tel: (812) 335-1653

Conceptual Architecture Action Guide

Purpose: 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. Key constructs are identified, including significant architectural elements such as components and relationships among them, as well as architectural mechanisms. Architectural mechanisms are designed to address cross-cutting concerns that cannot be localized within a single component.

Key Activities:

- Determine architecturally significant requirements.
- Create the high-level organizing structure of the system: identify architectural components, their responsibilities and relationships.
- Identify architectural mechanisms to address cross-cutting concerns.
- Validate that the architecture meets stakeholder goals, and where it does not, assess the impact.

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

From "Chapter 5: Conceptual Architecture". The book is titled *Software Architecture Action Guide,* by Ruth Malan and Dana Bredemeyer. *Look for it in 2005.* Please see our note about the book towards the end of this paper.

Introduction

In this Action Guide, we distill the essentials of the Conceptual Architecture phase of the Visual Architecting Process. The conceptual architecture collects together decisions relating to the key architectural constructs in the system.

Purpose of 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. Key constructs are identified, including significant architectural elements such as components and relationships among them, as well as architectural mechanisms. Architectural mechanisms are designed to address cross-cutting concerns (i.e., those not localized within a single component).

By focusing on key constructs and abstractions rather than a proliferation of technical details, conceptual architecture provides a useful vehicle for communicating the architecture to non-technical audiences, such as management, marketing, and many users. It is also the starting point for Logical Architecture, which elaborates the component specifications and architectural mechanisms to make the architecture precise and actionable.

The conceptual architecture diagram identifies the system components and interconnections between components, and the accompanying descriptions document the responsibilities of each component. Structural choices are driven by tradeoffs among interacting or even conflicting system properties or qualities, and the rationale section articulates and documents this connection between the architectural requirements and the structures (components and connectors, or mechanisms) in the architecture.

Inputs to Conceptual Architecture

Higher-level Architecture. If your organization’s enterprise architecture (EA) or product family architecture team has selected or created a Reference Architecture to be used across the enterprise, that is presumably referenced in your meta-architecture. At any rate, this Reference Architecture will form your starting layout for the architecture you are creating or updating. Yes, it will constrain your options as you decide on components and responsibilities, but it will also ensure consistency and leverage across the organization, and it can give you a head-start by eliminating a whole set of choices that the EA team has already weighed and ruled out. These are, afterall, the kinds of reasons that architecting is done at all.

Artifacts and learnings from the Init/Commit and Meta-Architecture Phases. The architecture vision, objectives, principles and strategies, key concepts and metaphors, and other models, decisions, and thinking from the preceding phases, as well as the requirements and other drivers that were input to those phases, are all input to the conceptual architecture phase.

Requirements. Many organizations have adopted the practice of assigning business analysts or requirements teams to collect requirements. It is unfortunate that this is generally done without (practically) any input from the architects, and the requirements specification is “tossed over the wall” at the architects. Nonetheless, if requirements specifications already exist, use them as your starting point. A *just-enough* process has no place for repeating work that has already been done!

Conceptual Architecture Activities

Conceptual Architecture Requirements: Establishing what is Architecturally Significant

If requirements specifications already exist, use them. That said, wherever we have influence, we strongly encourage architects to *partner* with business analysts or the requirements team, rather than following in their trail. Ultimately, the architect (or architecture team) should be responsible for the architecturally significant set of requirements.

We have to live in a world of compromise. No system can have ultra-performance, ultra-quality, ultra-scalability, ultra-you-name-it, and be cheap! By compromise we do not mean settling for the mediocre, but rather picking where to excel. We have to define where we will differentiate our system, and where we will accept competitive parity or “good enough” for our market segment(s).

Now, are your business analysts going to make these tradeoffs with keen insight into their implications? Are they thinking about this system, *and* its future variations? Are they thinking about what technology makes possible, and what it makes extremely difficult? If so, they are performing the up-front function of architects perfectly well, and you can leave defining the system qualities to them. If not, leverage what they are doing, and make sure that you understand enough about your customers, as well as your business strategy and organization's core competencies, to be able to define and prioritize the system properties (also known as system qualities or non-functional requirements) that are important to the market segment(s) you are addressing, and to your business.

Understand System Functionality. In meta-architecture, we collected just enough requirements to be able to make at least a preliminary but meaningful pass at scoping the system. Now we explore the functional requirements further, so that we understand sufficiently what functionality the architecture must support. We find *use case descriptions* to be a useful tool here, as they lend themselves well to communicating with stakeholders and to behavioral analysis in the definition and validation of the system architecture.

Capture System Properties. Architecture is all about making tradeoffs (or compromises) to achieve system goals as best we can. To do this, we have to understand what specifically we mean by the system properties and what achievement level will satisfice[1] our goals. Let us make this clearer with an illustration.

We want the software architect (or architecture team) be held accountable for the structural soundness of our software system, though of course many others will have a hand in creating the details of the structure. But what does “structural soundness” mean? Maybe we want it to mean “not brittle” where “brittle” means changes to the system are highly error-prone; or maybe we want it to mean “the structure is capable of accommodating load” where “load” may be transaction volumes within a certain range; or maybe we want it to mean it “will not yield to stress” where “stress” may be dramatic changes in scale; and so on. I'm sure you get the point—what we mean by structural soundness differs for each system. We must be as precise as we can be about what we mean for this system, and be clear about what is good enough.

We use a *qualities definition template* to guide us in making system properties unambiguous. Where possible, we quantify the system property goals, but some goals like “portability” are hard to quantify and we find *test cases* to be a useful tool for exploring just what we mean by the property and for providing a means to validate that the architecture actually meets the goal.

Understand Constraints. Understand the constraints you need to work within, including third party or legacy components and systems that are predetermined to be part of, or that must interact with, your sys-

1. This term is used in industrial engineering and decision science (introduced by Herbert Simon in his *Models of Man*, 1957) and means good enough but not necessarily the best.

tem. Also understand organizational constraints, such as resources and time available, as well as factors such as the capabilities of the development staff, being watchful of shortfalls in critical areas.

Determine What is Architecturally Significant. Given a set of requirements, or in the process of collecting requirements, the architect needs to assess what is architecturally significant, for these are the requirements the architect will focus on. Architecturally significant requirements are those that

- are representative: they capture essential functionality of the system (services the system *must* perform)
- have broad coverage: exercise many architectural elements
- challenge the architecture: identify issues/risks; highlight stringent demands on the architecture (e.g. performance requirements); are likely to change
- involve communication/synchronization with external systems

Conceptual Architecture Specification: Identifying the Major Elements

Review Proven Approaches, and Look For Opportunities to Create Architectural Advantage. Actively look for great ideas, and passé but tried-and-true ideas—review past architecture work and architecture patterns, and approach architect peers in your organization and "friendly" external groups such as suppliers or partners to share their strategies for addressing similar challenges to the ones you face. It is important to innovate, and you can buy time for such creativity by leveraging proven solutions. This is what is done in the automotive and other mature industries. In such industries, dominant designs have emerged that allow architects to focus attention on differentiation rather than the basic organizational structure of the system.

Take the trucking industry. Architects are not arguing about the basic organization, the systems, subsystems and components of trucks. But when NASA showed that a roof fairing on trucks would reduce drag over the tractor and improve fuel efficiency by some 20%, the industry was slow to move. Kenworth (we believe, and are working to validate this) introduced the rooftop fairing and for a while it cleaved the industry into the die-hard cowboys of the highway clinging to the fuel-guzzling box-shaped cabs of old, and those who were willing to risk ridicule in exchange for industry-reshaping fuel savings. Fuel savings won the day, and though others followed, Kenworth had strong early mover advantages.

Design the Organizing Structure of the System. During this phase we identify the major elements of the system. In doing so, we think about what responsibilities to put together in a component[1] and what to split apart. We think of alternative ways to decompose the system, and assess these alternatives against general principles of good architecture as well as the specific requirements and constraints, and earlier decisions, impacting our architecture. General architecting principles include divide and conquer, loose coupling among components and high cohesion within components, independence and encapsulation. Applying these principles helps us meet a number of system goals, such as understandability, and work partitioning so that teams and individuals can work more independently and make more compelling contributions by having a clear focus of responsibility.

These general principles and the specific requirements in our system, interact. For example, in a timing-critical imaging system, we need to weigh carefully whether to split apart probe hardware-related responsibilities from image processing responsibilities. If we put them together, we make the system vulnerable to high cost of change every time the hardware changes, because the impact of the change is not isolated. If we pull them apart, we face inter-component communication overhead each time these two components need to interact to achieve a piece of functionality.

We use a *(conceptual) architecture diagram* to show components and their relationships, enabling us to think about, discuss and document the high-level conceptual organization of the system. The component

1. These are conceptual components, not physical entities. The latter will be dealt with in the Execution (a.k.a. Physical) Architecture.

responsibilities are documented on the *component responsibility collaborator + rationale*[1] (CRC-R) *description* for the component. We add a rationale section to our template, reminding architects to record the rationale for the choices made in assigning responsibilities to this component and collaborators, providing a link to requirements, objectives and strategy. This helps make the experience and reasoning of the architect explicit for others to learn from. We also have a notes section on the template, to record forward-looking thinking that should be referred to in logical or execution architecture, or downstream during detailed design and development.

Create the Conceptual Design of Architectural Mechanisms. The system decomposition isolates chunks of the system and in doing so addresses some important properties (e.g., buildability and maintainability, portability, and reuse). Cross-cutting concerns are those that remain to be addressed across several, or even all, components. Identify which system properties are like this, and formulate mechanisms to address these cross-cutting concerns. Typical examples include security, system integration, session management, and configuration management.

One approach is to take each cross-cutting concern in isolation, first. This allows us to explore an idealized design until we bump up against the limits of technology and our own experience and innovative ideas. Of course, in the real world we do not get to focus exclusively on one system property. But this exercise allows us more creative freedom, and establishes what the limits are. We can then work on a solution that makes tradeoffs across the system properties, recognizing that we will have to make compromises to reach a solution that is good enough across the set of interacting, and sometimes conflicting, system properties.

Time is usually invoked as the argument against this approach, but exploring alternative approaches to addressing cross-cutting concerns at the conceptual design level is relatively inexpensive, especially when compared to the status quo approach of trial and error problem-solving during code development!

Often we need to explore the behavior of the system in designing mechanisms, and we find informal *component collaboration diagrams* or *sequence diagrams* helpful here. But remember that during conceptual design we are sketching out the approach, not pursuing the details of interfaces and protocols so we work with abstractions, not details like messaging syntax at this point. Of course, we add the caveat that if doing so becomes important for some reason, our iterative process allows us to take a temporary dive into the details of some aspect of the system before pulling back up to a higher level of abstraction and maturing the architecture at that level.

As we design architectural mechanisms we may find ourselves needing to refactor the components. We allow ourselves this freedom, until we are satisfied that we have taken all the paramount system capabilities (functions and properties) into account and we can explain and motivate how the architecture achieves them.

Document Assumptions, Assertions, Decisions, Issues and Ideas. A slew of architecture decisions are reflected in the conceptual architecture diagram and models of architectural mechanisms. These decisions are grounded in assumptions, experience, tradeoffs, and so on. Unless we document the reasoning behind, and rationale for, the decisions while it is all fresh in memory, we will become overcome by pressures of downstream tasks and the rationale will be lost. To put it bluntly, too often the only thing harder than getting the architects to write down their designs, decisions and rationale is getting the intended audience to read it! But that is no excuse. We have to create a record, for an undocumented architecture does not exist. You will not be able to communicate it, you will forget important details, and you will not be able to expand your capacity at critical points since you will have no mechanism for others to come up to speed on the architecture without getting direct access to your head![2]

1. This term is chosen to allude to and draw on the CRC (for "class responsibility collaborator") card body of practice that was established by Kent Beck and Ward Cunningham.
2. We're taking first-draft license here!

Conceptual Architecture Validation: Checking that the Elements Address the Concerns

The purpose of validation is to provide a forum to demonstrate that the architecture is good (technically sound) and right (satisfies stakeholders goals and concerns), and also to provide a venue to question and challenge the architecture. It is important to surface issues with the architecture as early as possible, while it is still relatively cheap to address these issues (at least, as expensive as it may be now, it will only get more expensive later). Furthermore, by this process we instill confidence that the architecture is good and right, putting us on a path to success.

Select Participants. We are all aware that too many dissenting voices can slow progress. At the same time, real issues need to be brought to light. Selecting the right people to be involved in the architecture validation exercise is a factor that requires thought. The following questions will help you to identify who you should involve in validation at this point in the maturation of the architecture:

- What concepts, approaches and decisions are being validated?
- Who has insight into those issues, can assess their goodness and rightness, and make suggestions?
- Who cares about the issues?
- Who needs to be shown how the architecture addresses their concerns?
- Who do you need to support and contribute to the process and the architecture?

Conduct Walk-throughs and Provide Reasoned Arguments. Walk through models of the architecture showing:

- how the stakeholder goals and concerns that were raised during visioning interviews and architecture requirements gathering are addressed.
- how the Meta-Architecture is reflected in the Conceptual Architecture. For example, illustrating how it embodies the relevant principles and is consistent with the style/patterns and/or reference architecture.
- how the architecture addresses concerns and goals raised during the validation process.

The validation process may bring new stakeholders into the discussion, and it may cause stakeholders who have already been involved to see their own goals or concerns in a new light. Of course we need to manage requirements changes, especially those that threaten our scoping decisions, but it is foolhardy to ignore new insights and opportunities to disambiguate requirements at this point.

Keep an Architecture Scorecard. We need to keep track of our progress against the goals set for the architecture. The *architecture scorecard* lists all objectives, requirements, and preceding decisions that the architecture needs to be evaluated against, and records how well we are doing against these objectives and success criterion. We also record what needs to be changed (requirements, the architecture, or existing projects impacted by the architecture), and assesses the importance and impact of these changes.

Conceptual Architecture Checklist. We also need to evaluate the architecture against the following criteria:

- Clarity: does each component have a clearly defined responsibility?
- Coupling: are there components with a surprisingly large number of interactions?
- Coverage: has all functionality been assigned to components?

Outcomes and Deliverables

We live in a world where tangibles—like deliverables—are demanded, but intangible outcomes are what fundamentally matter. Of course, architecture deliverables are essential to important outcomes. But the point is this: we need to think about what outcomes we want to achieve, so that we can shape expectations as to what deliverables we should produce. This allows us to focus on deliverables that will actually make a difference in achieving our desired outcomes.

Deliverables

As with the Meta-Architecture, and consistent with the IEEE 1471 Standard for Architecture Description, we need to consider the concerns of our primary stakeholders and tailor views that address specific concerns. We also need to tailor communication formats to match the communication styles and needs of the stakeholders. In the sections below, we cover deliverables that are generally useful.

Documents

We are often asked to provide a template for "*the* architecture document." If one is thinking of only one document, one must be thinking of the specification document. Indeed, the architecture specification document is an important tool, and the Visual Architecting Process provides the overall structure (Meta-Architecture, Conceptual Architecture, Logical Architecture, Execution Architecture, and Guidelines and Policies) as well as templates for the models and descriptions that make up each of these sections. But other documents are critical to reaching the different audiences. We know that market segmentation is important in product or service development. And we need to think about "selling" the architecture in more sophisticated terms. This is not to say we spend months on this marketing exercise, but we do need to tailor our message to the intended audiences or we will not have an audience—except by fiat.

Executive Briefing: Architecture Strategy. The audience for this document is primarily management, up to the level that is responsible for all systems impacted by the architecture. The briefing covers how we will deliver on the business and product strategy, producing differentiating capabilities or features. It also identifies concerns such as tough issues faced on similar past projects, or new risks accompanying new technical, market or organizational opportunities and directions, and articulates how we will address them.

Technical Briefing: Conceptual Architecture. The audience for this document is primarily the development or engineering community, and technical project management. We are so flooded with information in this "information age," that many of us shy away from reading in favor of "getting work done," even when we recognize that a bit of reading could help us get our work done more effectively. The goal of the technical briefing is to provide just enough reading to get the system view and essential architectural strategies across. Make the briefing interesting to the technologist, and brief. Identify the key tough problems the architecture addresses. Outline your technical approach to addressing these. Refer to white papers for details on technical approaches.

Architecture Requirements Specification. The specification document is the full reference work containing all requirements models, descriptions, notes, etc. which fully document and explain all decisions regarding system scope, architecture objectives, priorities, and architecturally significant requirements including functional requirements and characterizations of required system properties.

Conceptual Architecture Specification. The specification is your complete record of architecture decisions. For a system of reasonable complexity, this document (set) is already quite big, and you need to provide a means to navigate through it. Organizing this body of work according to the architecture decision model (meta, conceptual, logical, execution and guidelines and policies) provides for coherent sets of decisions at a consistent level of abstraction. The Conceptual Architecture Specification document, then, collects together all decisions, models, explanations, alternatives considered but rejected and why, and so forth, relating to the conceptual architecture. It covers how we have organized our system at a conceptual level:

- what are the components, what are their relationships and externally visible properties?
- what are the key mechanisms, how do they address the architecturally significant cross-cutting concerns, and how do they interact?

White Papers. The audience is the technical community, and the purpose is to explain and achieve buy-in. Write white papers motivating and explaining significant (sets of) decisions, such as the conceptual design of architectural mechanisms or key aspects of the system decomposition. To be effective at persuading the technical community, a white paper should clearly identify the problem that is being addressed, describe the solution and explain how it solves the problem, and outline alternatives that were explored and why they were not chosen.

Presentations

Documents provide a record and a reference, but presentations are essential to getting the architecture communicated and "sold." Use every chance, formal and informal, to present key aspects of the architecture to targeted audiences. And encourage others to do so. The more freely you allow others in your organization to present "your" architecture ideas, the more exposure those ideas will get. If others take ownership for those ideas, they are fully bought in to them. It is more helpful to see that as an achievement than as a threat.

Web Site

Make your architecture web site useful and it will provide its own draw. The great thing about the web is that it is right where the technical community lives, on their desktop. A well-designed web site serving content in a way that makes the right information package easily accessible on demand, is invaluable. It is definitely worth spending cycles architecting the information space covered by your various architecture documents, presentations, models and so forth. Make it simple to navigate to the right information for each (important) audience group—even to the level of individuals for those who are, or could be, influential in making the architecture soar to success.

Outcomes

A key outcome is confidence that "this can be done." We know where we are headed, and we have confidence we can get there. Specifically, the architect (or team of architects) is confident that there is good understanding of what capabilities the system needs to have, and these capabilities can be built in the planned timeframe with the allocated resources. Moreover, the architect has shared this path with management and key influencers in the development community, and gained their confidence by showing the approach that will be taken to build these capabilities, and articulating believable strategies to address the "gnarly" issues that beset such systems.

Other outcomes include:

- everyone is "on the same page:" there is a shared high-level understanding of the overall system (components and larger subsystems, their responsibilities and relationships), and the strategies (expressed as architecture strategies, principles, mechanisms, approaches, etc.) for building the key system capabilities.
- project management can use the Conceptual Architecture Diagram, and input from the architect(s) to plan the work allocations and detailed schedule. The project ramp-up varies according to the organizational complexities and architectural risks, mediated by practicalities like when developers are coming off other development projects.

Conclusion

As in any phase of the Visual Architecting Process, we collect sufficient requirements to make useful progress on system structuring and specification, and we validate that the work so far meets stakeholders goals or concerns. For conceptual architecture in particular, we need to understand enough about the system capability requirements to be able to think through and explain our approach to building these capabilities at a conceptual level.

Architects need to play a leading role in determining how the system will contribute to the competitve advantage of the business and where competitive parity is sufficient. The problems that we will have to tackle are determined during requirements definition. Architects need to keep one foot firmly planted in the business strategy space, as they help sort out and prioritize the architecturally significant requirements. That is to say, they need to be watchful of opportunities to differentiate yet cautious of risks including over-ambition and feature explosion. Architects need to be fully involved in establishing these requirements and they need to work with management to set priorities.

The conceptual architecture articulates a conceptual view of the system. It is analogous to the elevation and floor plan views that building architects use for their customers. In that paradigm, the blueprint adds the detail needed by various specialist contractors to perform their function in building the structure. Likewise, logical architecture adds the details that clarify the architecture, making it precise, unambiguous and actionable.

References

Bredemeyer, Dana and Ruth Malan, "The Role of the Software Architect," published on the *Resources for Software Architects* web site at http://www.bredemeyer.com/papers.htm

Clark and Fujimoto, "Power of Product Integrity", *Harvard Business Review*, 1990.

Malan, Ruth, and Dana Bredemeyer, "Less is More with Minimalist Architecture", IEEE's *IT Professional*, September/October 2002.

Malan, Ruth, and Dana Bredemeyer, "Architecture Strategy", published on the *Resources for Software Architects* web site at http://www.bredemeyer.com/ArchitectingProcess/ArchitectureStrategy.htm

Restrictions on Use

Acknowledgments

We would like to thank all the architects that we have worked with over the past decade for freely sharing their insights and hard-won lessons with us. We are most privileged to be able to pass on their learnings through our papers and workshops.

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.

We have other books in mind, but a distilled guide to our Visual Architecting Process is overdue! Up until now, only our clients have had access to the full Visual Architecting Process, largely through our training materials. This is because our writing projects have necessarily lagged behind the creation of products that keep Bredemeyer Consulting in business.

Brevity is a driving goal for this book. It has caused us to make choices, like pulling details on techniques and models out of the process overview and placing them in pithy “action guides”, one per technique, model or template. This means that if you are not familiar with a technique or model that we make reference to, you will have to wait with baited breathe for the related Action Guide to appear.

Joking aside, we look forward to your feedback, but please take into account that you don’t have all the pieces yet. We have chosen to put chapter drafts “out there” as quickly as possible so that you can have access to more information on the Visual Architecting approach, and we can have the benefit any input you are kind enough to take the time to give us.

Please join our mailing list to receive notice of new chapters and Action Guides 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

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

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 Classes:
London, UK on June 20-23, 2005
Indianapolis, IN on September 26-29, 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 Classes:
London, UK on June 20-23, 2005
Houston, TX on October 18-21, 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

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