Enterprise Architecture

Vol. 11, No. 10

Executive Report

CUTTER CONSORTIUM

10 Key Skills Architects Must Have to Deliver Value

by Michael Rosen, Director, Cutter Consortium Enterprise Architecture Practice

As the complexity of IT grows, more and more organizations are realizing the need for architecture. But the definition of what architecture is, the titles that architects have, and the role of an architect vary widely from one organization to another. Business, IT, management, and even architects don't necessarily know what a good architect does to add value in his or her organization. This *Executive Report* discusses the role of the architect and describes 10 activities that architects should perform to add value to projects.

Access to the Experts

Cutter Consortium is a unique IT advisory firm, comprising a group of more than 100 internationally recognized experts who have come together to offer content, consulting, and training to our clients. These experts are committed to delivering top-level, critical, and objective advice. They have done, and are doing, groundbreaking work in organizations worldwide, helping companies deal with issues in the core areas of software development and agile project management, enterprise architecture, business technology trends and strategies, innovation, enterprise risk management, metrics, and sourcing.

Cutter offers a different value proposition than other IT research firms: We give you *Access to the Experts*. You get practitioners' points of view, derived from hands-on experience with the same critical issues you are facing, not the perspective of a desk-bound analyst who can only make predictions and observations on what's happening in the marketplace. With Cutter Consortium, you get the best practices and lessons learned from the world's leading experts — experts who are implementing these techniques at companies like yours right now.

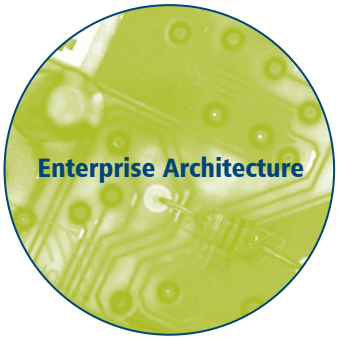
Cutter's clients are able to tap into its expertise in a variety of formats, including print and online advisory services and journals, mentoring, workshops, training, and consulting. And by customizing our information products, training, and consulting services, you get the solutions you need while staying within your budget.

Cutter Consortium's philosophy is that there is no single right solution for all enterprises, or all departments within one enterprise, or even all projects within a department. Cutter believes that the complexity of the business technology issues confronting corporations today demands multiple detailed perspectives from which a company can view its opportunities and risks in order to make the right strategic and tactical decisions. The simplistic pronouncements other analyst firms make do not take into account the unique situation of each organization. This is another reason to present the several sides to each issue: to enable clients to determine the course of action that best fits their unique situation.

Expert Consultants

Cutter Consortium products and services are provided by the top thinkers in IT today — a distinguished group of internationally recognized experts committed to providing top-level, critical, objective advice. They create all the written deliverables and perform all the consulting. That's why we say Cutter Consortium gives you *Access to the Experts*.

For more information, contact Cutter Consortium at +1 781 648 8700 or sales@cutter.com.



The Enterprise Architecture Advisory Service Executive Report *is published by the Cutter Consortium, 37 Broadway, Suite 1, Arlington, MA 02474-5552, USA. Client Services: Tel: +1 781 641 9876; Fax: +1 781 648 8707; E-mail: service@cutter.com; Web site: www.cutter.com. Group Publisher: Kara Letourneau, E-mail: kletourneau@cutter.com. Managing Editor: Cindy Swain, E-mail: cswain@cutter.com. ISSN: 1530-3462.*

Rob Austin · Ron Blitstein · Christine Davis · Tom DeMarco · Lynne Ellyn · Jim Highsmith · Tim Lister · Lou Mazzucchelli · Ken Orr · Mark Seiden · Ed Yourdon

Cutter Business Technology Council

10 Key Skills Architects Must Have to Deliver Value

THIS MONTH'S AUTHOR

Michael Rosen
Director, Cutter Consortium
Enterprise Architecture Practice

IN THIS ISSUE

It seems that nobody knows what an architect does. In this *Executive Report*, we look at this issue from two different angles. First, we look at common architectural titles and roles and describe what responsibilities we typically see associated with those roles across the industry. Then, we look at the skills that all architects have in common and describe 10 things that every architect can do to add value to his or her organization.

THE ROLE OF ARCHITECT

What does an architect do? Is it the same in your organization as in another? In my enterprise architecture (EA) practice, I see widespread differences in the titles, roles, and responsibilities of an architect. As an example, the titles of project architect, solution architect, software architect, and application architect are often used by different organizations to mean essentially the same thing. Typically, this role is responsible for the architecture and design of a specific project or application, usually specifying the logical structure and behavior of the software for that application in response to the functional requirements and perhaps specifying the infrastructure and platform required to meet the quality of service (QoS) (or nonfunctional) requirements for the project. (I will use the term "solution architect" when referring to this role.)

One of the critical concepts in this description is that of project, which leads to the first characteristic that distinguishes the role of the architect: *scope*.

Architectural Scope ... Project vs. Enterprise

Let's consider the scope of architecture and design (more on this in a minute). In Figure 1, the middle level portrays the project-level architecture that we have been discussing. For example, an *n*-tiered, Web-based application architecture is represented by the "W" architecture. For that architecture, there are potentially many different conforming designs, which are shown on the lowest level of Figure 1 as W1-W*n*.

As an aside, my definition of design is that it is the specification of the structure and behavior of a specific

system. Notice that this is pretty much what I described the role of a solution architect to be. Architecture on the other hand is intended to define what is common across multiple systems. In other words, architecture is for many like things, whereas design is for one specific thing. So it turns out that architects, especially at the project level, are often involved in creating design, not architecture. Rather than creating architecture, they are applying it at the project level. Now, let's get back to scope and Figure 1.

The scope of application architecture in an enterprise embraces many different application styles. The figure shows two application architectures (though of course an enterprise would likely have more than two) labeled E and W. These might describe how to build applications using enterprise application integration techniques and how to build Web applications.

At the enterprise level, the goal is to understand and facilitate the integration and coordination of all these application styles. The EA at the top in Figure 1 presents a higher level of architecture that describes how all of these applications fit together to meet enterprise goals and contribute value to the overall enterprise.

In other words, architecture is fundamentally about commonality: common processes, common information, common infrastructure, and common standards, all intended to meet specific business and technology goals, as well as reduce and manage complexity in an ever-changing environment.

Given this definition, the first step in the process of architecture is to figure out what must be common. This is driven by the business goals and requirements and the alignment of IT systems in order to meet those requirements. Unfortunately, often the requirements themselves are not clear. In this case, the initial job of architecture is to determine and clarify the goals and strategy. This is often the first task of business architecture.

Once the requirements are determined, we can proceed with defining what must be common in order to meet them. For example, if the business goal is to have a single customer view across multiple lines of business (LOBs) in order to achieve increased customer penetration, then the definition and information about a customer must be common across those lines of business. Perhaps the billing process needs to be common across all the LOBs, a consolidated bill needs to be sent to the customer, and so on.

So now the architecture must include the specification of commonality; that is, the semantic definition of the common customer. This requires digging into the next level of detail. For example, obtaining and using a single customer view probably involves new business processes, the aggregation and normalization of data from multiple sources, the integration of many applications, and the creation of new customer information services, in addition to provisioning the infrastructure to support this.

Architecture must determine how to address all of these aspects of the solution, and that will require a variety of

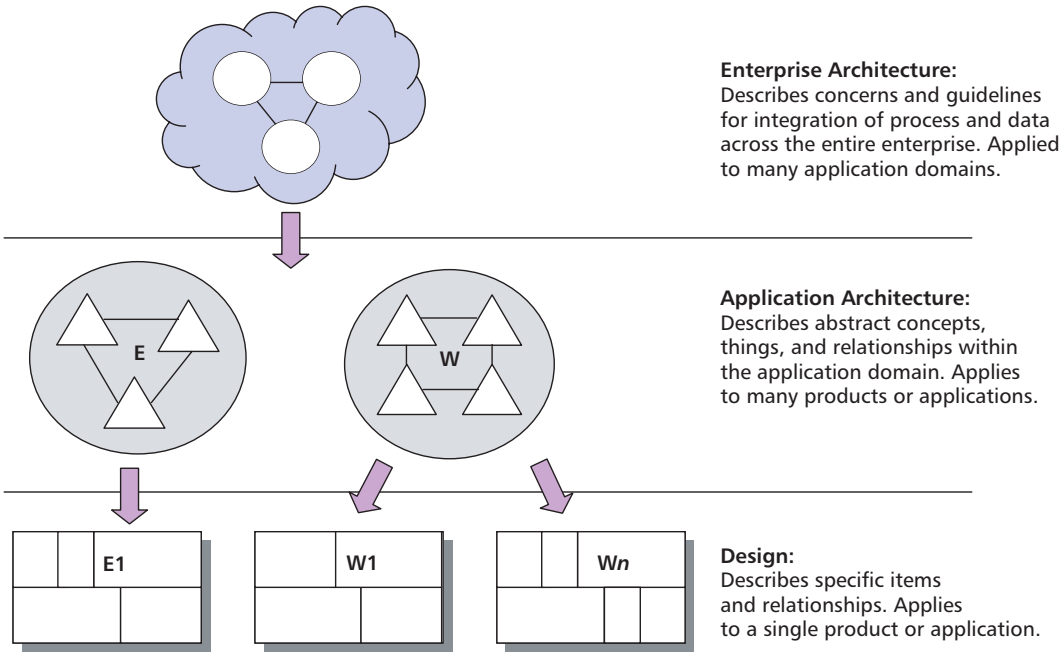


Figure 1 — Architectural scope.

architectural artifacts across architectural domains. Here again, the scope of the architecture is important. At the enterprise level, the concerns will be related to the overall enterprise, or collection of applications and processes. At the project level, the concerns will center on applying the enterprise standards and reference architectures to a specific project.

So architectural scope determines the role and responsibility of a particular architect, as often reflected in the architect's title. An enterprise architect is responsible for designing the solutions and standards for those things that must be common across all LOBs and applications in order to meet the enterprise goals. The enterprise architecture establishes the context that applications need to be designed within. Hence, solution architects are responsible for more than just the design of a solution; they are responsible for ensuring that the solution fits within the context established by the enterprise architecture.

The solution architect acts as a bridge between the design of the specific application or solution and three important areas:

1. Business architecture of the application domain
2. Internal architecture of the application suite (application architecture)
3. Enterprise architecture (particularly the EA business and technology architectures with which the application needs to align)

The following job description from one of my clients sums this up nicely:

The solution architect:

- Develops and maintains the application architecture for specific business functional areas in compliance with the enterprise architecture.
- Provides oversight on the architecture and design of an application within a line of business.
- Participates in assignments concerning development of target conceptual and logical architectures for applications.
- Participates in project and design reviews to evaluate and ensure that the design being applied meets policies, principles, and standards.
- Reviews and aligns product choices to avoid redundant effort and ensure that enterprise architecture standards are met.

While the solution architect is typically working in the application domain, this is not always the case. So to really know what an architect does, we need to look at another distinguishing characteristic: the subject area or *architectural domain*.

Architectural Domains

Architectural frameworks give us a conceptual model that we can use to frame the different aspects or domains of enterprise architecture and their relationship to each other. There are many frameworks for enterprise architecture such as Zachman, TOGAF, FEAF, and DoDAF. Figure 2 shows a very simple conceptual model for enterprise architecture.

The "BDAT" framework shows the primary architectural domains of business, information (or data),

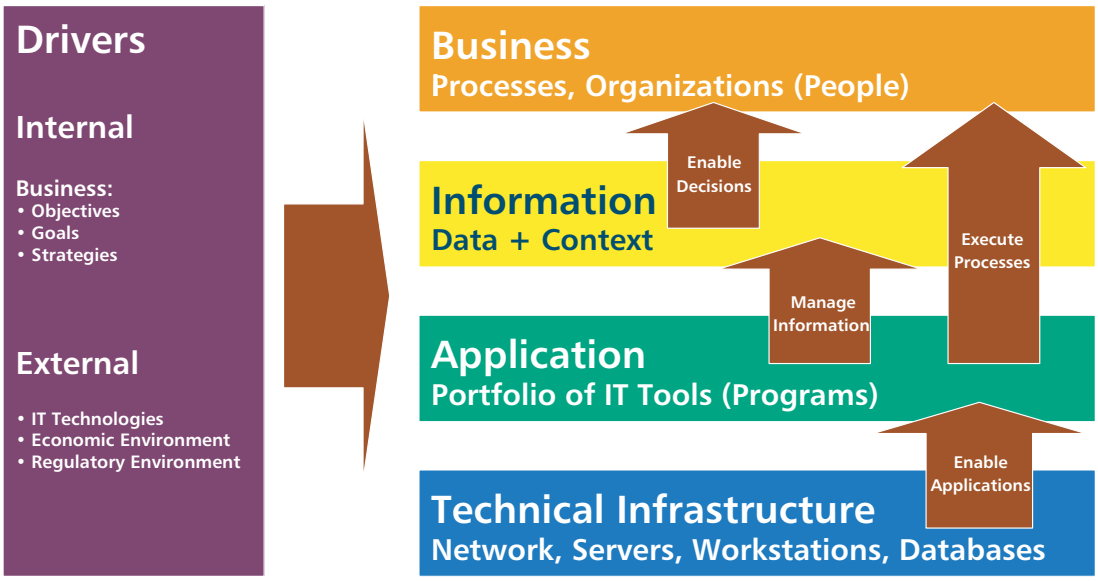


Figure 2 — BDAT architecture framework.

application, and technology, and the architectural disciplines of security, methods and tools, and people and organization, where the disciplines apply across all of the architectural domains. Almost all EA frameworks have these four basic domains.

Each domain describes a particular aspect, or subject area, of the overall enterprise architecture and has specific relationships to the other domains. For example, we see that information enables decisions at the business level and applications enable processes at the business level. Applications also manage information. Finally, technology enables the applications (which manage information and processes, which enable the business).

For each architectural domain, there is a set of concerns, goals, or concepts associated with that subject area. Each set can be described by a conceptual model and perhaps documented in a commonly used formal model. These models define the vocabulary used within that architectural domain and role. Together with the domain, we factor in the scope of responsibility of a particular architect. Specifically, the concerns at the enterprise level are different from those at the project level. This is true for all of the architectural domains. Let's briefly examine each of the primary architectural domains noting the concerns at both enterprise and project levels.

The information architect is concerned with providing a managed information environment for operational and transactional data.

Business

The business architect is concerned with defining the business so that IT systems can effectively support it. The enterprise business architect wants to help achieve alignment, but alignment of what? The architect defines business goals, strategies, and outcomes as targets for alignment, and tactics, organizational structures, and initiatives as ways of carrying them out. These are defined in such models as business value chains, Business Motivation Models, organization charts, and roadmaps.

At the project level, business architects are trying to achieve alignment of business requirements with systems implementations. One of the primary methods for doing this is modeling business processes and business rules. Processes are shown in business process models where architects are concerned with breaking down the overall process into tasks and decisions that are performed by business workers or actors. Processes deal with business entities and use business documents to pass information.

Information

The information architect is concerned with providing a managed information environment for operational and transactional data and for transforming that data into information to support business analysis and reporting. At the enterprise level, the architect wants to provide a consistent view and usage of operational data across multiple applications and to rationalize data and information storage to minimize duplication and simplify access. Like all architects, the information architect is interested in commonality and specifically in providing a common mechanism for moving and transforming operational data into analytical data, sometimes called a data flow architecture, such as the one published by Cutter Fellow Ken Orr. Data transformations should be based on common business and information models. The information models are typically created as entity relationship diagrams (ERDs) where the important concepts are enterprise entities and the relationships and constraints associated with them.

At the project level, the information architect is concerned with information that has a more limited scope. Access to and use of the information is based on business rules and is governed by security and privacy requirements of both the enterprise and the application. A data model describes the application-level information, which is likely to be different from (but related to) the common enterprise information model. A transformation between the application and enterprise models is required and might be the responsibility of the enterprise- or application-level architects, or both of them.

Operational information for the application is defined in ERD models. Analytical information for the application, often accessed through a data mart, is typically defined in terms of a multidimensional data model. Here the concepts are the main transactional facts and the associated dimensions of analysis.

Application

The application architect is concerned with commonality in applications. At the enterprise level, this means creating reference models and standards that specify a common structure or architectural style that promotes sharing of common responsibilities, using common services in a consistent fashion, supporting a common user interaction style and configuration mechanisms,

using a standard technology platform, having common management, monitoring, and operations procedures, and so on. This is not done in an attempt to limit the creativity of application developers (as many will argue), but to improve integration between applications, allow for sharing of common information, have consistent results for the same operation no matter how it is performed, and reduce the cost and complexity of maintenance and enhancements.

To achieve these goals, the application architect needs to first specify the architectural style to be used and the specific roles and responsibilities of the architectural elements that make up that style. This is the place where technology aspects such as performance, scalability, reliability, security, and so on, should be factored into the reference architecture, not on each individual project. The application architecture can be expressed as a conceptual drawing, but it should also be formally specified in a reference model created in UML. A set of patterns is often developed as an aid to implementing the reference architecture. The architect also needs to create standards, guidelines, and templates that describe how specific aspects of the application development are done. For example, how is the logging service used? What constitutes an error, warning, informational, or debug message? What set of common error codes will be used across all applications? One of the best ways of getting people to follow these guidelines is to provide examples or reference code that they can adopt and apply.

At the project level, the application architect (or solution architect) is concerned with applying the enterprise context (reference models, patterns, standards, guidelines, templates) to a specific project. The architect acts as a bridge between the enterprise and the application. This can be an area of contention with the project team, because it is not the team's responsibility to understand the enterprise context. The team is responsible for delivery of projects. It is the responsibility of the solution architect to help the project meet its requirements in a way that conforms to the enterprise application architecture, so some influencing skills are in order.

Technology

In the technology domain, the enterprise technology architect is responsible for providing common platforms that support the different application architecture styles (a few, one hopes) with the appropriate QoS. Technology architecture often includes everything but the kitchen sink, such as systems, storage, security, networks, data centers, management, capacity planning, performance analysis and monitoring, disaster recovery/business continuity planning, and who knows what else.

At the project level, the technology architect is tasked with provisioning a specific instance of the standard platform for the specific application and integrating it into common management, security, disaster recovery/business continuity planning, storage area networks, backup, services, and so on. In addition, when projects or applications have requirements that are not met by the standard platform, the architect must create an appropriate solution that meets the project needs and fits into the rest of the technology infrastructure.

All domain architects act as the bridge between the specific project and the enterprise context.

Commonality

Again, at the project level, all domain architects act as the bridge between the specific project and the enterprise context. Of course, architects have other concerns; this is not an inclusive list by any means, and what any individual architect does is likely to vary between different organizations. As well, any given individual architect may have several roles. In a smaller organization, a single architect may do all of them; in a larger enterprise, multiple architects may share a single role.

However, there is a common pattern that is important to notice and understand. For each particular architectural role, there is an associated set of concerns. Some kind of conceptual model can describe each set of concerns and specify each in a commonly used formal model. These models define the vocabulary needed for the specific concern and role. For example, the "enterprise business architect" (role) defines "business goals and strategies" (concerns) using the concepts and specifics of "value chains" (conceptual model) and "Business Motivation Models" (formal model) to define "strategies, objectives, and tactics" (vocabulary).

As we said before, every enterprise is somewhat different, and the titles, roles, and responsibilities are far from consistent across the industry. Nonetheless, we provide some guidelines in Table 1. The different architectural titles (roles) are listed in the left-hand column. For each role, we list some typical responsibilities, concerns, and outputs of that role depending on the architectural scope — if they are working at the enterprise scope or at the project level.

Table 1 — Architect Roles and Responsibilities

Title	Architectural Scope	
	Enterprise	Project
Enterprise architect	Understands the big picture and relationship between domains.	May consult at the project level.
Solution architect Project architect	Does not work at the enterprise level, but must understand the EA context and provide feedback on how EA actually works from the project perspective.	Acts as a bridge between enterprise context, business domain, and the project design.
Business architect	Business goals, strategy, alignment using value chains, Business Motivation Model, context models.	Business requirements specified as processes and rules in Business Process Modeling Notation.
Information architect	Enterprise semantics and information model. Data flow architecture, enterprise data warehouse.	Data models in entity relationship diagrams, database schema, and analytical models.
Application architect	Reference architectures, common services, patterns, frameworks.	Logical structure and behavior of application in conformance with EA in UML. (At the project level, application architect often is the same as solution architect.)
Technology architect	Common technology platform, storage, networks, data center, quality of service.	Implementation of project on common platform.

10 THINGS AN ARCHITECT DOES TO ADD VALUE

Regardless of the specific role, we can start to see a common set of skills that architects must have, such as having a big-picture view, analytical approach, abstraction, conceptualization, and modeling. Let's look at how these skills can be applied by every architect to add value to his or her organization.

All Architecture Is Local

Cutter Senior Consultant Jeroen van Tyn likes to say that "all architecture is local." What does that mean? It means that no matter what the architecture is from the enterprise perspective, or in terms of architectural standards or governance, architecture is effective only when it is actually applied locally at the project level.

In other words, value does not come from the creation of architecture. The world's best architecture provides no value if it isn't used. Architectural value comes only when it is applied. In general, the opportunities to apply architecture occur with projects.

In fact, one of the major challenges that architects face is getting involved with the business and with projects. Often, we must look for project opportunities to provide architectural value. If we understand the unique skills of an architect, we can see how to use those skills to assist projects. In this section, we describe 10 things an architect does, in the context of the general lifecycle of a project. For each item, we will describe the overall principles involved and then illustrate them with an example. These examples demonstrate the architectural activity and principles but are not the only way to achieve them.

We can organize the architect role in terms of three broad categories of project involvement:

1. Requirements elicitation and analysis
2. Solution design and specification
3. Influence

The 10 things an architect can do to add value fall into these three areas, as described below.

Requirements Elicitation and Analysis

At the beginning of a project, the issues are around correctly understanding the problem and requirements. Here, architects are able to use their discovery and analytic skills to bring out requirements and to describe the problem space in a clear and unambiguous matter. In this phase of a project, architects inquire, integrate, and analyze.

1. Inquire

Projects are initiated with the goal of solving specific problems. Getting to the core of the problem and soliciting requirements is the first step in addressing any given set of requirements. Of course, the requirements are often vague and presented in the limited focus of a specific application domain. So the inquiry must solicit both specific requirements and goals, as well as an understanding of how those requirements fit into a broader context (such as the enterprise). This comes about through discussions and questions of the project stakeholders.

An important line of inquiry is to question assumptions. This is a key activity and responsibility for an architect. Far too often we accept assumptions, whether they are explicitly stated or, more likely and more dangerously, unstated assumptions. Groundbreaking products, those that routinely "change the game," do so by questioning assumptions and getting beyond those that no longer apply.

Recently, in an exercise on teamwork and team building, my class was subjected to the famous "prisoner's dilemma." The catch of the exercise (and the dilemma) is that for your team to "win," you have to let the other team win. If you try to beat the other team, both lose. Only through cooperation does the scenario turn into win-win. As is usually the case, the teams figure this out too late to change their behavior, which is based on an unstated assumption that the objective of the exercise is to defeat the other team. In the postexercise debrief, one of the students put it this way: "There was an abundance of unstated and unexamined assumptions." Exactly!

Sometimes we talk about this as "thinking outside of the box," or getting beyond established assumptions to see which ones do and don't apply to the current scenario. So, next time you're analyzing requirements and solutions, make sure to look for and challenge both the explicit and implicit assumptions.

2. Integrate

Architects act as a bridge between a given project or design and how that project fits into the broader context. One of the major benefits that an architect brings to the enterprise is integrating the solution for the particular project with the business domain, enterprise concerns, industry standards, established patterns, and best practices.

I use what I call architecture-driven design. Each architectural domain (shown in rectangular, shadowed boxes in Figure 3) is driven by a set of enterprise requirements (shown in oblong boxes). This sets the overall environment into which the project needs to be integrated.

Each project also has its own set of unique requirements, those that we helped elicit in skill 1. However, the application does not exist by itself, but rather it exists in the context of the overall enterprise. So we use the architectural domains of business, information, application, and technology as a starting point for the analysis and design of the project's problem space. Notice that it is the enterprise architects who create the overall context and the solution architects who apply it to the specific project.

We also want the business processes, services, and information required and supported by the application to conform to the business and data architecture of the enterprise.

Because we want to have consistency of applications across the enterprise, we want the applications to conform to the application architecture and to run on the infrastructure defined by the technology architecture. We also want the business processes, services, and information required and supported by the application to conform to the business and data architecture of the enterprise.

These architectures provide an enterprise context within which the analysis and design are performed (box in the middle right of Figure 3). In this example, design leads to services based on developing service-oriented architecture (SOA). This is done following a technology-independent approach. When it is time to implement the service, the implementation architecture describes how to do so using a specific technology choice (described by the technology architecture), taking into account tools, governance, and development processes.

After implementation, the service will be deployed into production. The operational architecture (often part of technology architecture) describes operational aspects, such as monitoring, logging, and backup, as well as deployment aspects, such as replication, configuration, and topology.

Enterprise architecture provides a context, set of patterns, and platform to support applications. These are required to meet the overall enterprise goals of cost, consistency, and flexibility of IT systems that meet

business requirements. Each architectural domain provides a specific context to inform the overall solution design. In addition, application-level patterns provide common solutions to problems that span applications. The solutions architect integrates all of these areas together during project analysis and design.

3. Analyze

Now that the project requirements have been collected and integrated into the overall context, an architect has to analyze the information. The analysis includes answering three architectural questions:

1. What are the important elements of the problem or solution?
2. What are the relationships between them? And, how do the relationships describe the behavior of the overall system?
3. How do the elements and relationships combine to provide value higher up? How do they serve the purpose of the system versus the purpose of an individual part?

Often, analysis will be broken into two important parts: a model of the problem and a model of the solution. The first is used to precisely describe the requirements within the broader perspective and without any bias toward how IT might implement it. The solution model then describes (at a high level) how the problem will be solved in terms of IT concepts, services, systems, and applications.

An architect has specific analytical skills in terms of collecting, integrating, and organizing concepts into a clear, precise, and unambiguous presentation. These are often skills that are not as highly developed in the business analyst, and therefore this situation provides an ideal opportunity for architecture to engage with the business. I like to use the business context diagram, as shown in Figure 4, as a tool to enable conversation with the business.

We know that the business operations inside and outside of the enterprise are made up of interactions and exchanges of information between parties. To describe the overall set of interactions, we use a business context diagram. The context diagram includes the major parties, represented by the rounded rectangles, and the

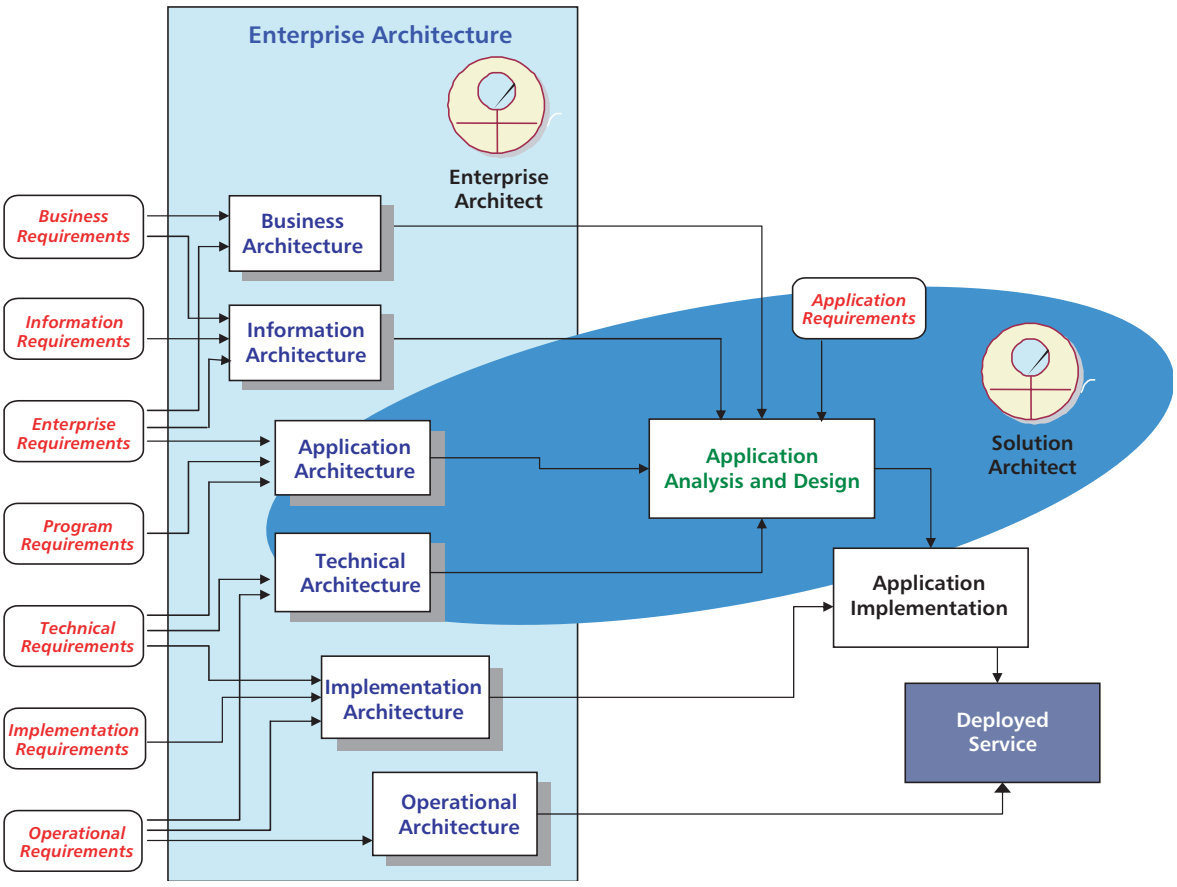


Figure 3 — Architecture-driven design.

messages that they exchange, represented by the arrows. You create the context diagram by talking with the business analysts and walking through all of the interactions required for the end-to-end capabilities of the project.

The context diagram is made up of the following semantic elements:

- **Actors.** The main parties of the interactions (the rounded rectangles). Typical actors are people, organizations, or systems.
- **Messages.** Information exchanged between actors (the arrows). Messages are typically documents, packages, and electronic communications.
- **Subjects.** The business matters that the messages are about. The subjects are not explicit in the drawing but are implied by the interactions and messages. Subjects are typically products and services.

You can think of a shipping package, such a box from a bookseller, as a metaphor for these elements of the context model. The package has a shipping label with "from" and "to" addresses. These are the actors. The package box itself (to which the labels are attached) is like the message. It goes between the "from" actor and the "to" actor. The contexts of the box are the subject. They are what the message is about.

For example, in Figure 4 the customer and storefront are actors. The customer places an order, which is a message. The subject of the message (and hence the order) is books that the customer wishes to purchase.

Notice what a business context diagram provides:

- **Overall interaction.** The context model represents the overall interaction of all aspects of the system. It is purposely kept at a high level and includes only business concepts, no technology. It is a combination of all the different business scenarios and transactions. Any single scenario represents one path through the overall diagram (a subset of functional areas and messages). The context diagram is the first place where we can start to identify commonality in function and information.
- **Shared information.** The messages describe the information that must be shared and exchanged between parties to complete the different transactions. It does

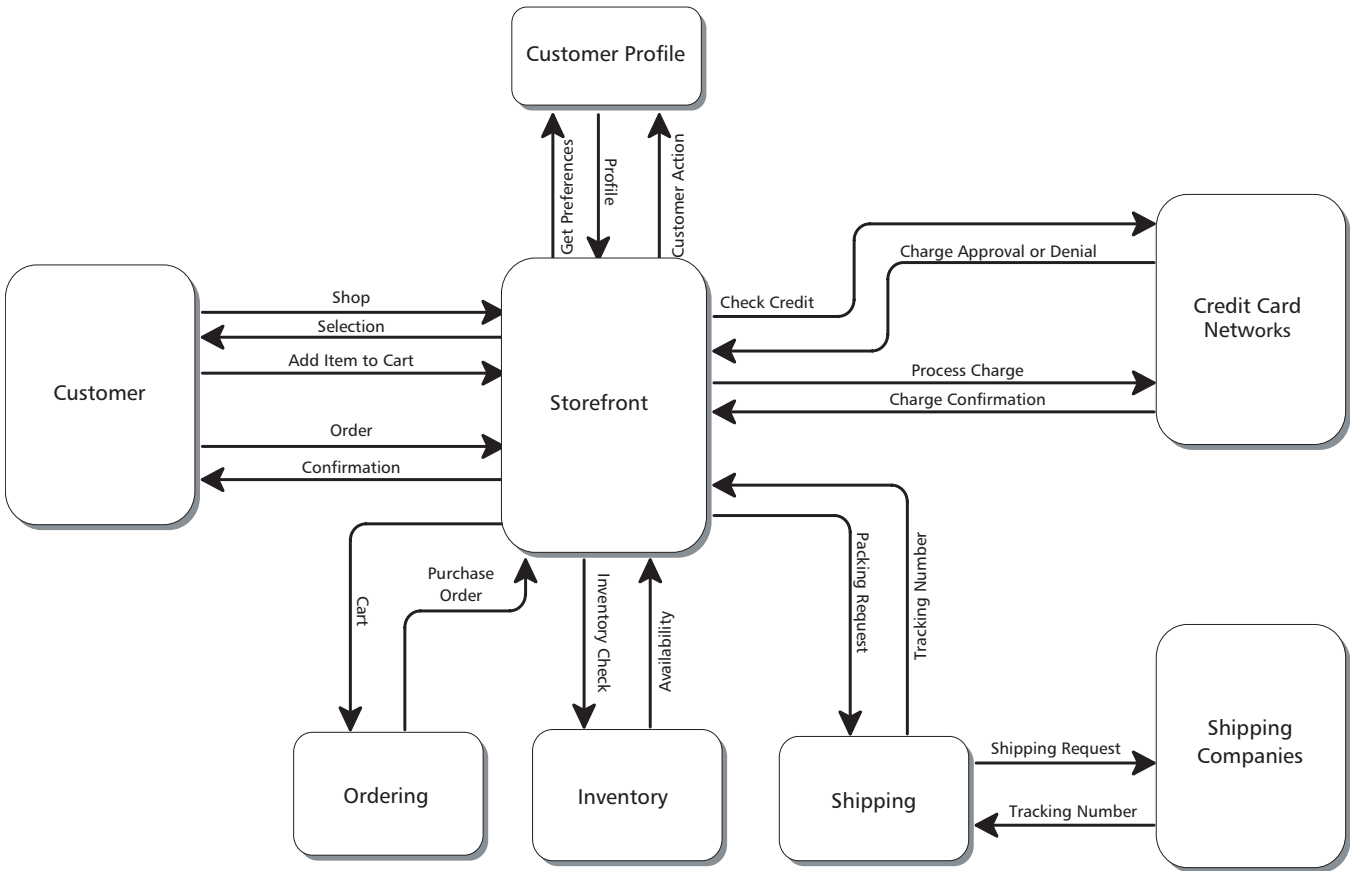


Figure 4 — Sample business context diagram.

not describe the details of any information within the different functional areas, but only the information exchanged; that is, shared. Notice that this is exactly the information that you need for the semantic information model and to design the service interfaces.

Business context diagrams provide an excellent communication mechanism with the business. They are intuitively understandable and nontechnical. They focus on business concepts. The process of creating them helps to bring a common understanding to the different parts of the business.

Almost invariably, the business analysts learn something useful in the process of helping to create the context diagrams, and they use them in the future to better understand their domain. In other words, the diagrams provide value to the business analysts by describing their business domain in a fashion that is clearer, more complete, better integrated, and easier to understand than they had before. Your next interaction with the business analysts will be much easier because now they will view architecture as a value-adding activity, rather than as something that is time consuming.

Of course, the context model is just one example of an analysis artifact and activity. The key concept here is the application of an architect's analytical skills to provide clarity and value to the business and project.

Solution Design and Specification

Moving right along with the project, now that the problem is clearly understood, it is time to turn attention to the solution. Here, the architect extends his or her analytical skills and focuses on clearly and precisely describing the design of the solution, starting at a conceptual (high) level and working down to the details. In this phase of a project, architects conceptualize, abstract, visualize, and formalize.

4. Conceptualize

Once the overall, integrated problem is analyzed, the architect needs to create a conceptual vision of the solution. This can be in the form of a conceptual architecture diagram, a drawing that shows the major users/channels of the system, the other systems it has to interact with, and the major logical functions and data that it must perform or use. The diagram establishes the scope of the project within those pieces. Figure 5 illustrates a sample conceptual model for a pharmaceutical project.

The project is to build a system for testing new drugs. There are two major parts to the system: one part is used to design the tests and the forms and processes for collecting test data, and the other is used to execute the tests and collect the data. This is the first of several projects that will implement an overall service-oriented

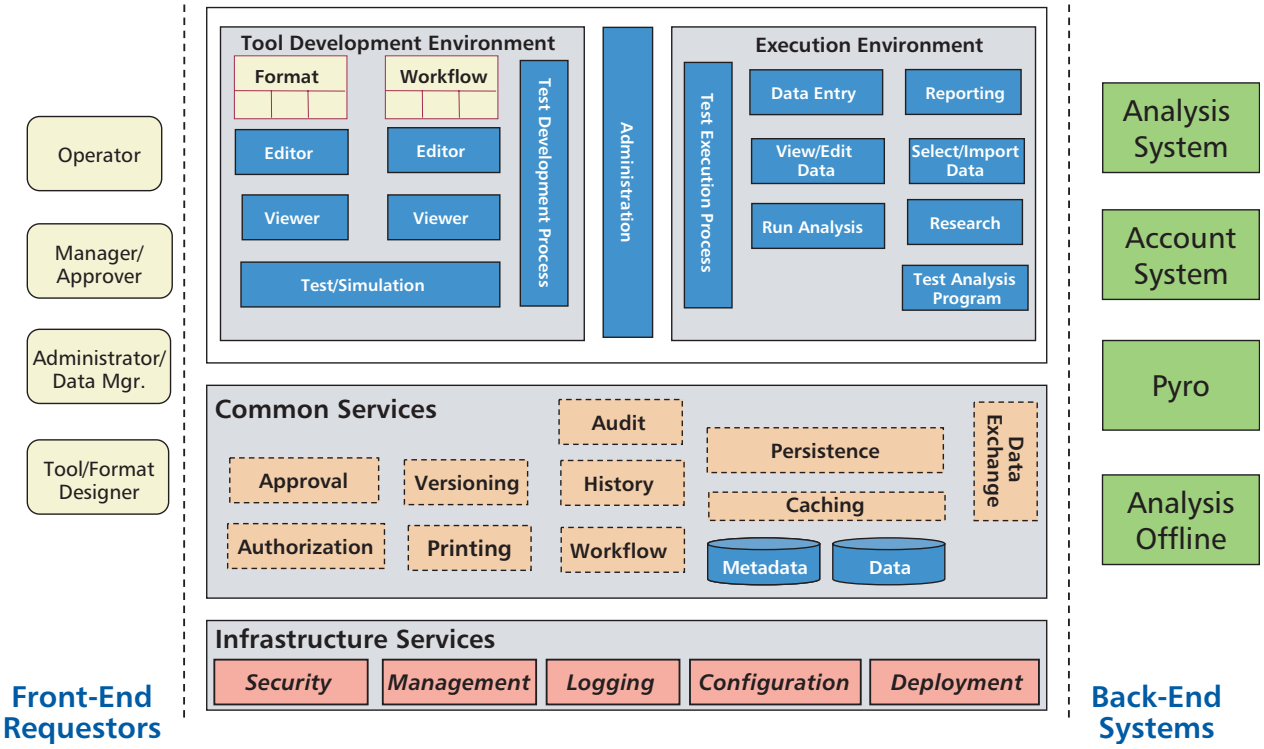


Figure 5 — Conceptual architecture.

environment, so a major concept of the solution is the idea of common services, such as approval, auditing, and versioning of test designs.

In the conceptual architecture drawing, the major users of the system are shown on the left (operator, manager, administrator, and designer), and the major systems that the project needs to interact with (analysis, account, and pyro) are shown on the right. The scope of the project is shown in the center between the dotted lines. The major subsystems are the tool development environment, execution environment, and common services. In addition, some infrastructure services are illustrated to highlight the difference between typical technical services and new, business-oriented SOA services.

The conceptual architecture serves to communicate the overall concepts to a broad audience. It describes:

- Interactions, channels, and systems
- Scope of the project
- Major information concepts
- Primary functions
- Overall structure of the solution in its environment

As we develop the drawing to include these concepts, we need to also answer some other key questions:

- Who is the main audience for the drawing?
- What concepts are in the drawing? What concepts are not illustrated?

5. Abstract

In answering these questions, the architect has to use the skills of abstraction. Abstraction can be defined as the suppression of irrelevant detail. The biggest challenge is to determine what is relevant and what isn't.

One way to begin is through the use of architectural viewpoints, such as business, information, application, and technology perspectives. When we determine who the main audience for the drawing is, this gives us a big hint as to what types of information are going to be important. So one key abstraction is to apply separation of concerns to establish (filter) what details are and are not important for that viewpoint.

Within each perspective, the viewpoint will also be presented in different levels of abstraction, often referred to as conceptual, logical, and physical architectures as shown in Figure 6 (not all perspectives will have all of these levels). This requires another, more subjective, and perhaps more difficult aspect of abstraction. How do we take a bunch of more detailed, lower-level components or concepts and combine them into a single, more abstract, higher-level concept? Well, we ask some fundamental questions, such as:

- How are the lower-level concepts similar (structure, purpose, behavior, interactions, etc.)? What are the key characteristics that they have in common?
- How are the lower-level concepts different? What differences are important or distinguishing?

At this point, experience plays a big role. The more times we have applied abstraction, the easier it is to recognize and apply common patterns.

Figure 6 shows the typical levels of abstraction, from conceptual, which is the most abstract, to logical, to physical, which is the least abstract. The process of refinement is how we go from a more abstract to a less abstract level, where refinement is the addition of specific detail. Like abstraction, the challenge is to understand what specific details should be added. A key goal is to have a consistent level of detail within any given model. If the details that we add are consistent with the architectural perspective, then we are truly doing refinement.

However, this is a good time to point out some confusion that is often associated with the model of abstraction shown in Figure 6. Often, the models associated with the names "conceptual, logical, and physical" are not only levels of abstraction but are also different perspectives. For example, the conceptual model is drawn from the business viewpoint, the logical model from the application viewpoint, and the physical model from the technology viewpoint. Although the typical logical model is usually somewhat more refined than the

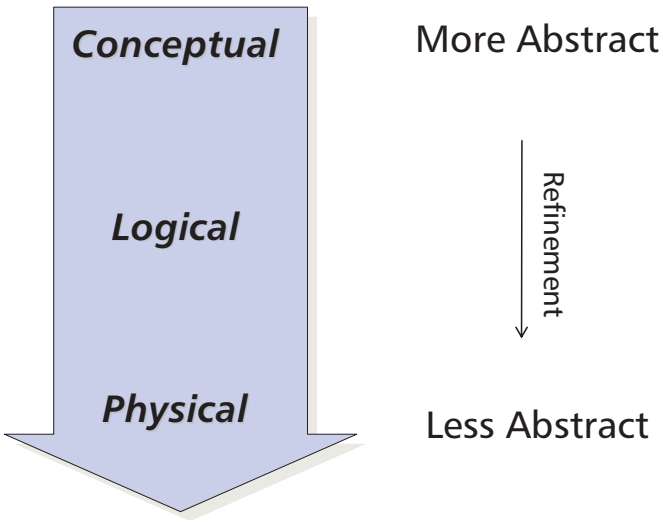


Figure 6 — Levels of abstraction.

conceptual, and so on for the physical, the major difference is viewpoint, not abstraction.

6. Visualize

They say a picture is worth a thousand words. It is also an excellent way to represent the architectural drawings and models at each level of abstraction. So another key skill and function of the architect is to create visual representations of the different abstractions and viewpoints.

Figure 7 shows an example of a conceptual application architecture. It illustrates the logical structure of the application, described by a set of patterns that define an *n*-tiered architectural style. The architecture is shown as an informal drawing, which illustrates three important concepts of the structure of an application:

1. **Architectural elements** — specific roles and responsibilities of the structural elements that make up the application
2. **Distribution tiers** — the logical assignment of elements to distributed systems
3. **System layers** — the separation of functional elements to isolate changes in technology

Each white box in Figure 7 represents an architectural element in the construction of an *n*-tiered application. It is a technology-independent unit of application responsibility. Each gray box in the figure represents a service that is applicable across all the application tiers and is common to the infrastructure or all applications. The architecture has been defined to support a range of enterprise requirements, such as:

- Distribution
- Scalability
- Technology independence
- Device independence
- Application integration
- Separation of application development from infrastructure development
- Future enhancements and migrations

The details of the architecture are described in my earlier Cutter *Executive Report* "Enterprise Architecture by Example."[1]

Another word of caution: in general, architecture is represented by models, and most models are visual. But be aware that not everyone is necessarily a visual thinker. As you convey the important architectural concepts, other forms of communication may also be necessary.

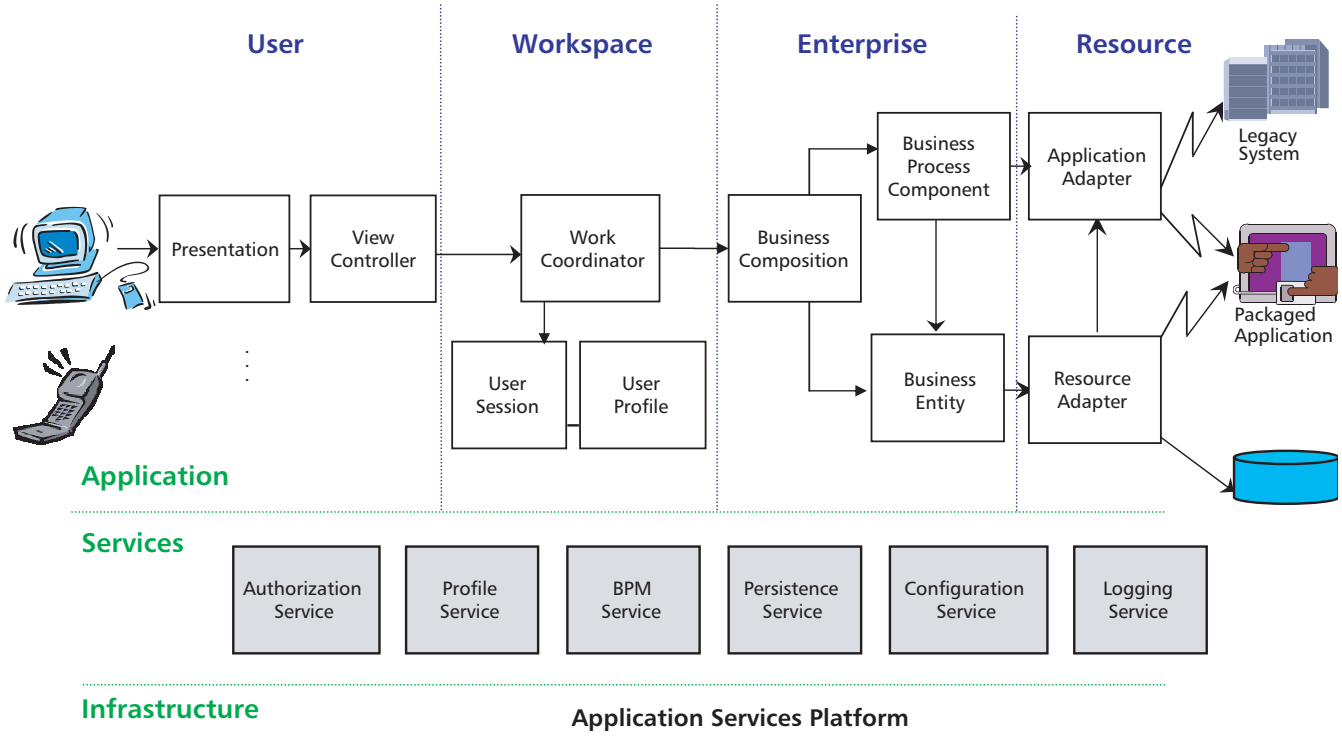


Figure 7 — Conceptual application architecture visualization.

7. Formalize

As we pointed out earlier, Figure 7 is an informal drawing. Of course, architecture needs to be more than just pretty pictures. It needs to be specific enough to unambiguously communicate the details to anyone who implements the architecture. It also needs to be complete and precise enough to allow a design to be evaluated for conformance.

An architectural "specification" is the usual approach to formalization. But the specification does not necessarily have to be a document. A formal visualization in the form of a complete and precise model, expressed in industry-standard notation, may often be preferred because a formal model can be implemented and enforced within a modeling tool or design framework. The formal specification may be called a reference architecture or architectural metamodel.

Figure 8[2] illustrates a fragment of the formal architectural reference model associated with the *n*-tiered architecture of Figure 7. It defines the specific roles, relationships, and constraints for each of the architectural elements illustrated in the conceptual architecture. It is one of several drawings that together completely describe all of the different architectural elements, their roles, characteristics, constraints, and interactions. Together, these models answer the key architectural questions of: What are the main elements? What are their relationships? How do these relationships describe the behavior of the system? How does the system, the elements together, provide overall value?

Not all architectural formalisms are reference architectures — in fact, most are not. Think back to the question of architecture scope. At the enterprise level, we want to specify the architecture to provide the context for project development. This context can be formalized as a reference architecture, such as is shown in Figure 8. But, at the project level, we will create analysis and design models that apply that context to the specific requirements of the project. Those models also need to be formal, complete, precise, unambiguous, and represented in industry-standard format.

This is where the vocabulary, concepts, and models of the different architectural domains come in. For example, business models may describe business processes in Business Process Modeling Notation, whereas information models may be represented as ERDs, and application models documented in UML.

The key value is the formalization of the model using a format that is appropriate for the domain; that is, one easily recognized and understood by practitioners of that domain.

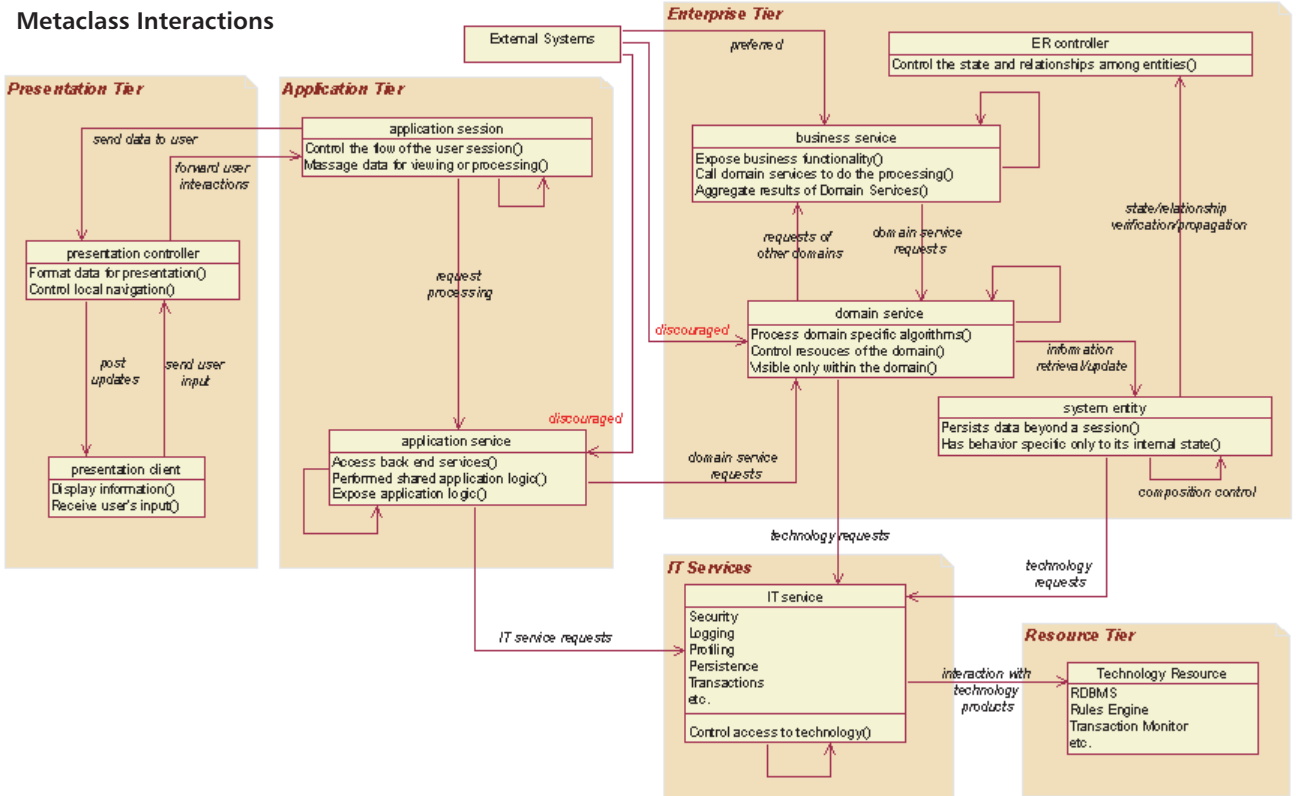


Figure 8 — Reference architecture metamodel. (Source: Terry Merriman.)

Influence

Once the solution is designed, the architect generally is not involved in the implementation of the project. The architect is still involved in seeing it to completion, making sure the design really works, providing guidance, and validating that it remains true to the design and architecture, but he or she probably isn't writing code. (This is my opinion. See the sidebar "Should an Architect Write Code?") At this point, architects are involved in communication, enabling, and assisting.

8. Communicate

This is probably the single most important aspect of an architect's job. Fundamentally, architects are in the role of communicator. After they establish and formalize a solution, they need to communicate that solution as well as its importance and value throughout the organization. Communication skills include:

- Ability to write
- Ability to speak

SHOULD AN ARCHITECT WRITE CODE?

The debate constantly comes up about the role of an architect and whether or not he or she should write code as part of a project team. This can become a somewhat heated discussion with lots of strong opinions, but I think it ultimately comes down to the typical consultant's answer: "It depends." It depends on your environment, what you mean by an "architect," and what you expect the architect to accomplish.

For example, the job of the enterprise architect is to understand things at an enterprise scope, beyond any individual application or line of business. Architects must understand how all similar applications fit together, what must be common across all of the applications to meet enterprise goals (such as single sign-on, common customer profile, standard server platform), and what each application must do to support the overall enterprise. The enterprise architect must also understand the relationship between the different architectural domains (business, information, application, technology, security, and so on) and how to achieve the enterprise goals in terms of the different domain architectures. So the questions I ask are: how does writing code help the enterprise architect achieve this, and just as important, is it the best use of the architect's time?

More relevant to the discussion is the role of the solution architect. The solution architect acts as a bridge between the areas of:

- Enterprise architecture (particularly what the application needs to align with)
- Domain architecture
- Internal architecture of the application

In other words, we do not expect the project team to be knowledgeable and current about EA; that is not their responsibility. The team is responsible for executing a project. Instead, it is the job of the solution architect to understand EA and to convey the important aspects (important with respect to this project) to the project team so that the team is incorporated into the design and implementation of the project.

How does the solution architect bridge these different perspectives to effect an architecturally conforming implementation? The critical part of this is the architect's ability to convey the concepts to the team, get them incorporated into the design, and make sure they make it into the implementation. This will depend on the skill of the architect, the type and size of the organization, and what the architect is expected to accomplish. Here, personal experience is likely to influence your opinion.

There is no doubt that a solution architect must be conversant in both design and implementation skills and be able to talk the talk with the project team. Let me give a few examples from personal experience. At a large insurance company, I helped build a team of 50 application architects, most of which were developers in the past. This company does about 2,000 projects a year, for which the most important 500 get application architects assigned to them. You do the math: that means 10 projects per year for each architect. Most architects can do two projects concurrently, which gives them 10 weeks (at half effort) for each project. In this time, they are able to work with the application team to create a good, EA-conforming, high-level design. Later on, they participate in design and implementation reviews. The architects simply do not have the time to write code, yet in the four years since we initiated the program, it has proven to be very effective — so much so that project teams compete to get an architect assigned to their project.

On the other side of the argument, Scott Ambler, a great architect for whom I have the highest regard and respect, believes equally strongly that an architect must write code to be effective. In his environment — working with smaller, agile teams — his experience is that the architect has to be able to get his or her hands dirty in the code to be effective.

So it boils down to your environment and your expectations of an architect. No one answer is right for everyone, but that won't stop us arguing about it.

- Ability to draw
- Ability to empathize with your audience
- Ability to facilitate

Almost always, architects act from a position of responsibility without direct authority over those who would use the architecture. Thus, the only tool available is to influence potential users by explaining how it is important and in their own best interest to follow the architecture.

One of the keys to effective architectural value is in the phrase above: *their own best interest.* There are two important aspects to this. First, we need to be able to communicate the value of architecture. Second, and perhaps more important, architecture must be designed so that it supports and adds value to the expected users. Think back to the example of the business context model in skill 3, analyze. The business is happy to participate in the architectural activity because the architecture provides value to it.

9. Enable

Even the best designed, formalized, and communicated architecture may not be successful. The equation for architecture value is actually pretty straightforward. If using architecture will make someone's job easier, they'll use it. If it adds extra steps and work, without adding extra value, it will be ignored. Of course, achieving this goal is anything but simple, but a key to achieving architecture's goal of influencing IT projects and systems depends on the extent to which architects enable the target audience to easily use the architecture.

In other words, the success of architecture will be judged by its adoption. It should be easier, faster, and result in more success to complete a project using the architecture than without it. Remember that the value from architecture does not come from creating it, but from applying it. Therefore, the focus of the architecture group should be outreach: to help projects use and conform to the architecture not to create architecture.

Several enabling activities make an architecture more valuable. These include:

- **Consulting.** Providing advice to project teams about architectural requirements, design, and so on. Helping to resolve issues that have a cross-project or enterprise scope. Helping with the solution to particularly difficult problems. Conveying the value and importance of architecture.
- **Education.** Providing training about the purpose, meaning, and value of architecture. Describing the specifics and details of particular aspects of the architecture. Providing and describing a methodology for implementation of the architecture at the project level, focused on the particular domain audience.
- **Guidelines.** Providing specific guidelines for architectural requirements and implementation.
- **Examples.** Providing code examples of conforming implementations. This is one of the best ways to effect adoption. Software developers are generally happy to include already developed and well-written code into their projects. Provide code examples for security requirements, single sign-on, use of the auditing service, and so on.

Large architecture documents have a funny way of collecting dust on the shelf.

- **Documentation.** Providing well-documented architecture. This includes both the conceptual and formal drawings as well as any written documents. However, be aware that the document will at best be used as a reference. Large architecture documents have a funny way of collecting dust on the shelf.
- **Checklists.** Providing specific checklists of architectural requirements that will be validated during the review process. Forearmed with this list, projects will be able to meet the expectations and requirements of the governance and review process much easier.

Unfortunately, we can't get rid of the review process altogether. But by adopting the enabling activities listed here, we can make it much less of a burden.

10. Assist

Finally, one of the primary enablers for architecture is to actively assist projects in using it. This is the single most important activity architects can do to make their architecture real. This is essentially the role of the solution architect. Virtually all successful architecture programs include some aspect of consulting to projects.

Figure 9 shows a sample organizational structure for an architectural group. The team is shown on the left of the drawing. Typically, it is part of a centralized IT function. There are three important aspects to it:

1. **Governance/architecture review board.** It is the responsibility of the architecture group to manage and monitor compliance with the architecture. This is typically done through governance processes and the

review of projects by the architecture review board. As mentioned above, a program that focuses on enablement will have a much easier time with governance compared to a program that focuses on compliance and review.

2. **Core EA team.** This team of architects creates the enterprise specifications, models, standards, and frameworks that describe the enterprise architecture and provide context at the project level. There is usually someone with the role of chief architect who is responsible for understanding the overall picture and the relationship between all the architectural domains. Each domain has architects who specialize in that domain. Depending on the size of the group, a single person may handle multiple domains, or a given domain may have several people who specialize in it. In this case, there is usually a lead architect for the domain.
3. **Consulting architects.** This team is assigned to assist projects. The architects can specialize in any particular domain and are assigned to projects based on their strengths. Typically, a consulting architect will have a primary domain where he or she is especially strong and a secondary domain where he or she is knowledgeable but perhaps not expert. Rarely do we find architects who are strong in three or four domains.

On the right side of Figure 9 are the lines of business. Often, they are a separate business unit from the central IT function. They may or may not have their own IT capabilities, but often will have their own business architects. While the LOB architects are experts in their business domain, they are not responsible for the overall enterprise architecture. It is still the responsibility of the consulting architects to bridge these areas at the project level. Projects are run by the LOBs, where the consulting architects assist in the analysis and design of the solution.

CONCLUSION

The list of skills is not complete, but I hope it gives you some food for thought. Is this what you or architects at your organization do? Are you missing or skipping some important steps? Are you spending time on the right activities that bring value to your organization?

Although there is little consistency in architectural titles and roles across the industry, there are many common skills that all architects can apply to add value. In this report, we determined that adding value with architecture means applying it at the project level. To do this, architects must be practical and balance short-, medium-, and long-term goals. They must communicate the value and goals of architecture to the project and sponsors.

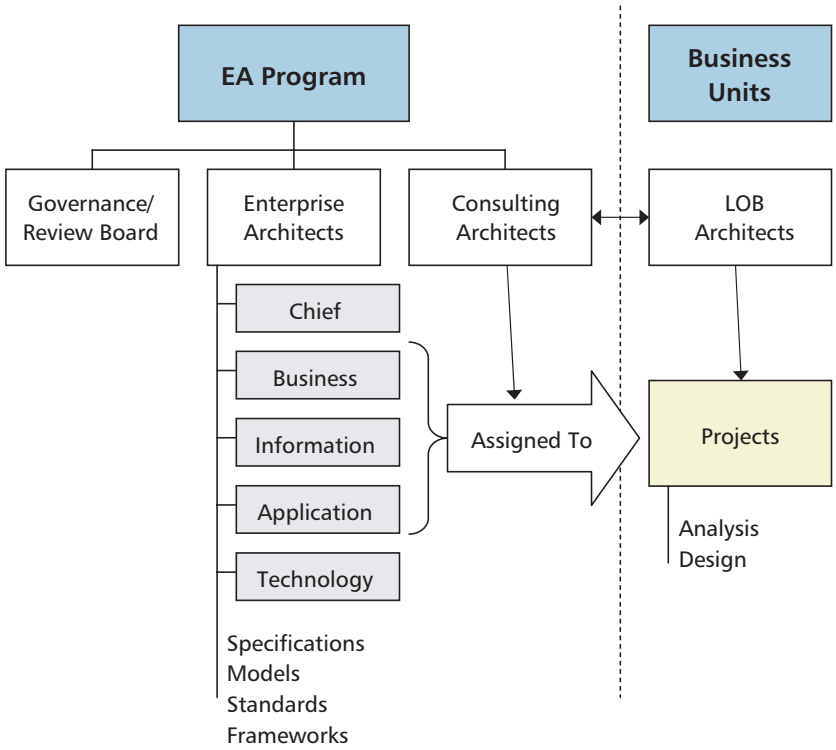


Figure 9 — Sample organizational structure for an architectural group.

They must also make it easy to apply the architecture. Those responsible for defining the architecture have a critical responsibility to enable its adoption through training, guidelines, frameworks, and examples. The most effective way to get projects to adopt architecture is to have architects help in the analysis and design of projects.

Architecture is about understanding what must be common across a set of things. In other words, architecture specifically takes a big-picture view of things. When architects work at the project level, their responsibility is to make sure that the project fits into the overall big picture of the enterprise in terms of business alignment, information semantics, application styles, common infrastructure, and so on.

To do this, architects apply a variety of techniques. Inquiry skills, and in particular questioning assumptions, allow architects to get to the heart of requirements. The basic principles of separation of concerns and abstraction help the architect analyze the requirements and begin to conceptualize the solution. Then, increasing levels of detail and formality are applied to create conceptual and formal models of the solution. This requires a different way of thinking than might be typical of someone dedicated to a specific project. The architect must think more comprehensively and formally and understand the interdependencies of the project within the whole enterprise. The architect applies enterprise context, industry standards, patterns, and best practices to arrive at the best solution for a specific project.

When discussing the role of an architect, one client remarked that an architect is "someone who tells you what you can't do." Besides not being in the category of things that add value, that stereotype of architects is certainly not how I want to be perceived. The list of 10 items presented in this report focuses on the architect as someone who applies his or her skills to help people and projects make the right decisions and do the right things. Architecture groups want to be seen as delivering services and adding value to projects and the enterprise, not as getting in the way. Although governance is often part of an architect's responsibilities, it's not the part the architect wants to be known primarily for.

Finally, one of my friends that manages an architecture group commented that there were 11 things that his architects do. I left out number 11: whine and complain. Oh well, nobody's perfect.

ENDNOTES

[1]Rosen, Michael. "Enterprise Architecture by Example." Cutter Consortium Enterprise Architecture *Executive Report*, Vol. 11, No. 5, 2008.

[2]Rosen, Michael, and Terry Merriman. "An Application-Centric Approach to Understanding Architectures." Cutter Enterprise Architecture *Executive Report*, Vol. 6, No. 12, 2003.

ABOUT THE AUTHOR

Michael Rosen is Director of Cutter's Enterprise Architecture practice and Senior Consultant with its Business-IT Strategies practice. He has more than 20 years' technical leadership experience architecting, designing, and developing software products and applications. Currently, he provides expert consulting services in the areas of EA, SOA, and MDA. Previously, Mr. Rosen was CTO at M2VP, Inc., a consultancy concentrating in IT architecture, where he developed the company's practice area using MDA to specify and implement EA. He also was Chief Enterprise Architect at IONA Technologies, PLC, where he was engaged in the development of the overall product architecture for IONA's next-generation Web services platform and in the creation of the reference architecture for building applications on that platform. Prior to joining IONA, Mr. Rosen was Chief Enterprise Architect at Genesis Development Corp., where he provided architecture consulting on large-scale applications and infrastructure for Global 1000 clients in insurance, finance, and telecommunications. While at Genesis, he led the development and formalization of a generic enterprise architecture and software development practices used throughout the company. Before joining Genesis, Mr. Rosen was a product architect, technical leader, and developer for numerous commercial middleware products for vendors, including BEA and Digital. His involvement in product development includes Web services, Java, CORBA, COM, messaging, transaction processing, DCE, networking, and operating systems.

Throughout his career, Mr. Rosen has been a frequent technical speaker and author. He has presented keynotes, tutorials, and seminars at conferences in the US, Europe, and Asia and has articles in publications that include *Cutter IT Journal*, *Web Services Journal*, *EAI Journal*, *Software Magazine*, and *Enterprise Development*. He is coauthor of *Applied SOA: Service-Oriented Architecture and Design Strategies, Developing E-Business Systems and Architectures: A Manager's Guide* and *Integrating CORBA and COM Applications*, which grew out of his contributions to the OMG COM/CORBA interworking specification. He can be reached at mrosen@cutter.com.

About the Practice

Enterprise Architecture Practice

Today the demands on corporate IT have never been greater. Cutting costs and accelerating time to market for individual line-of-business projects are still priorities, but even that's not nearly enough anymore. Companies are now looking for strategies to better leverage their entire IT infrastructure. They want IT to deliver sophisticated enterprise applications that can provide value across many lines of business and provide marked differentiation from their competitors. The Enterprise Architecture Practice provides the information, analysis, and strategic advice to help organizations commit to and develop an overarching plan that ensures their whole system fits together and performs seamlessly.

The Enterprise Architecture Practice offer continuous research into the latest developments in this area, including Web services, enterprise application integration, XML, security, emerging and established methodologies, Model Driven Architecture, how to build an enterprise architecture, plus unbiased reports on the vendors and products in this market. Consulting and training offerings, which are customized, can range from mapping an infrastructure architecture to transitioning to a distributed computing environment.

Products and Services Available from the Enterprise Architecture Practice

- The Enterprise Architecture Advisory Service
- Consulting
- Inhouse Workshops
- Mentoring
- Research Reports

Other Cutter Consortium Practices

Cutter Consortium aligns its products and services into the nine practice areas below. Each of these practices includes a subscription-based periodical service, plus consulting and training services.

- Agile Product & Project Management
- Business Intelligence
- Business-IT Strategies
- Business Technology Trends & Impacts
- Enterprise Architecture
- Innovation & Enterprise Agility
- IT Management
- Measurement & Benchmarking Strategies
- Enterprise Risk Management & Governance
- Social Networking
- Sourcing & Vendor Relationships

Senior Consultant Team

Our team of internationally recognized specialists offers expertise in security issues, e-business implementation, XML, e-business methodologies, agents, Web services, J2EE, .NET, high-level architecture and systems integration planning, managing distributed systems, performing architecture assessments, providing mentoring and training, overseeing or executing pilot projects, and more. The team includes:

- Michael Rosen, Practice Director
- Paul Allen
- Douglas Barry
- Dan Berglove
- Max Dolgicer
- Don Estes
- Pierfranco Ferronato
- Clive Finkelstein
- Michael Guttman
- David Hay
- Tushar K. Hazra
- J. Bradford Kain
- Bartosz Kiepuszewski
- Sebastian Konkol
- Jean Pierre LeJacq
- Arun K. Majumdar
- Thomas R. Marzolf
- Jason Matthews
- Terry Merriman
- James Odell
- Ken Orr
- Wojciech Ozimek
- Jorge V.A. Ronchese
- Oliver Sims
- Borys Stokalski
- John Tibbetts
- Sandy Tyndale-Bisco
- William Ulrich
- Mitchell Ummel
- Jeroen van Tyn
- Jim Watson
- Tom Welsh
- Bryan Wood