CHAPTER 1

Software Architect Capability Model

Many information technology (IT) and business professionals often fail to provide clear answers to these three fundamental questions: What do software architects do? What artifacts[1] do they deliver? How should architecture skills be assessed, quantified, and vetted?

At a first glance, these sound like easy queries to address. The conventional notion that a software architect fulfills the same duties as a building or landscape architect is utterly incorrect. There is no parallel between these two occupations, because they exercise different practices in distinct fields of expertise. Furthermore, they are commissioned to achieve dissimilar goals.

A software architect is required to perform a vast number of activities, typically handled by more than one professional. So, is it possible to deduce from these tasks what architects actually do or what they deliver?

In the context of this chapter, the simplest answer we offer to such challenging questions is this:

A software architect does what a specific organization needs—nothing more!

[1] Software architecture artifacts are various deliverables produced during a product development life cycle. They describe the internal and external architecture of software. They typically include strategy documents, technical specifications, design blueprints, data models, security models, deployment and integration charts and diagrams, best practices and standards documents, and more.

This assertion is deliberately too broad. This concept affirms, however, that a software architect must respond to *business and technological requirements* of a particular organization. In other words, architecture tasks and deliverables vary from one institution to another. Moreover, while working for different lines of business, architects seldom tackle the same challenges, nor do they always provide solutions for similar problems.

This chapter, therefore, offers a simple architect capability model with step-by-step instructions—assisting individuals and organizations to answer these three important questions:

Occupation What does a software architect do?

Deliverables What should a software architect deliver to provide potent organizational solutions?

Capabilities How should software architecture talents be vetted, assessed, and quantified to ensure successful facilitation of enterprise projects?

Software Architect Capability Model: Benefits

The software architect capability model can be leveraged by organizations and individuals to promote business and personal professional agendas. When it comes to fostering enterprise business strategies and missions, the offered architect capability model will provide limitless opportunities for business growth. The model will become a potent platform for project improvement and a tool for recruiting exceptional architecture talents, subsequently minimizing enterprise expenditure.

Individuals who aspire to become software architects will find the capability model a powerful tool for career change and professional promotion. For those already actively pursuing the architecture practice, the model can boost their aptitude to provide robust remedies for organizational challenges.

How Should Organizations Utilize the Software Architect Capability Model?

Onboarding IT personnel is utterly costly. Not only does the interviewing process consume human resources, but also candidate vetting typically puts strain on employees whose daily schedules get disrupted. But the chief challenge is even harder: many organizations do not utilize any methods to evaluate candidates' skill sets. Moreover, there is no benchmark or assessment methodology in place to quantify interviewees' knowledge of and capability to perform the jobs they are applying for.

Furthermore, to a large extent there is no industry-wide model that organizations can leverage to ensure that a software architect talent will indeed contribute

to a specific enterprise project. Put differently, there is no method in place to map a software architect's capability to facilitate enterprise imperatives. The consequences of screening failures are typically dire: allocated budgets evaporate quickly, and returns on investment never materialize. Another concern to contemplate is dwindling or inadequate institutional technical knowledge. This can hinder the fulfillment of organizational strategy, vision, and mission.

To address these enterprise concerns, consider the following list. It summarizes the organizational benefits when constructing software architect capability models.

Hiring process Improving the vetting mechanisms of candidates to enable hiring the best possible talent in the marketplace

Project management Delivering powerful business solutions by assigning adequate architecture skills to a project or any other enterprise initiative

Organizational knowledge base Maintaining a robust organizational knowledge pool by retaining the most experienced workforce

Why Create a Personal Software Architect Capability Model?

It's not just enterprise managers who are hiring who should figure out how software architects ought to promote organizational business goals and strategies. Architects, too, should individually be motivated to ascertain what their personal contributions should be to an organization or an industry.

By creating a personal software architect capability model, individuals will be able to *assess* their competence strengths and weaknesses in the space of software architecture. Then they can further leverage the model's findings to *augment* their knowledge for the purpose of honing the craft of their occupation.

Another compelling reason for creating an individual architect capability model rests upon the fact that useful organizational solutions are always delivered by software architects who are fully aware of their professional capabilities.

The list that follows, then, reflects the notion that *personal goals should be intertwined with organizational imperatives—neither could survive without the other*.

Ascertaining personal knowledge gap Revealing what additional skills a software architect would need to become more instrumental when providing solutions to organizational problems

Preparing for job interviews Constructing a personal architect capability model would divulge what type of technical and/or business skills are necessary to obtain certain architecture positions

Planning for career opportunities and job promotion Assisting with establishing a sound career path for pursuing higher-level positions in organizations

Rudimentary Guiding Principles

While constructing the software architect capability model, as guided in this chapter, either for personal or enterprise needs, adhere to these essential principles:

It's all about the "what" and the "how." As mentioned in the introduction of this chapter, an architect capability model addresses these simple questions: *What do software architects do? What do they deliver*? *How should their skills be vetted and assessed?*

It's all about delivering solutions. Software architects are hired to provide *solutions* to organizational problems.

It's all about teamwork. Architects do not operate in a vacuum, nor are they employed to pursue their personal agendas without benefiting the enterprise's strategies and vision. They must collaborate with business and IT professionals to deliver potent remedies for arising organizational challenges.

Software Architect Capability Model Creation Process

The list that follows summarizes the process for creating a software architect capability model. For each of the items in the list, a corresponding section in this chapter meticulously discusses the building block of every step of the way.

Step 1: Provide requirements and specifications. This section offers insight about the necessary requirements and the role they play in driving architecture solutions.

Step 2: Identify software architecture practices. This section elaborates on the practices segment of the capability model. It conveys how vital architecture occupations are for meeting business requirements and technical specifications.

Step 3: Establish software architecture disciplines. The architecture disciplines portion of the capability model defines areas of knowledge, fields of expertise, and specialties that a software architect must possess to provide effective solutions for business and technological imperatives.

Step 4: Add software architecture deliverables. The deliverables segment of the capability model identifies the required architecture artifacts associated with each discipline.

Step 5: Quantify skill competencies. This part of the capability model depicts an architect's level of aptitude to deliver valuable artifacts for a project or any other enterprise initiative.

Requirements Drive Architecture Solutions

Deeply rooted in almost every product development life-cycle methodology, requirements are being delivered in response to organizational imperatives. Some requirements aim to fulfill business vision, mission, strategies, and even marketing endeavors. Others are designed to address business challenges related to market competition and survival.

The software design work that an architect provides has a staunch correlation to particular problems that requirements seek to address. Therefore, architects must meet requirements to provide tangible organizational solutions.

A software architect capability model must be driven by requirements. *Without requirements, solutions could not be provided. Lack of solutions, consequently, may expose a business to inordinate risks*.

The sections that follow, therefore, address three fundamental questions:

Events What are the chief events that trigger the issuance of requirements?

Entities What are the organizational entities chartered to deliver requirements?

Requirements What type of requirements are necessary for constructing useful architect capability models?

Requirements Issued by Problem and Solution Domain Entities

In almost every corporation there are two different subject-matter expert (SME) groups, responsible for addressing internal and external organizational challenges and unforeseen events that can hamper business operations.

Problem domain Typically affiliated with the business unit, performing a variety of tasks, such as risk analysis, business analysis, product management, business requirements, business strategy, business architecture, marketing, etc.

Solution domain Characteristically a part of an IT organization that employs architects, developers, technical writers, operation personnel, cybersecurity experts, and others

How Do the Problem and Solution Domains Collaborate?

There are innumerable business adversities that an organization must tackle—for example, loss of revenue, increased market competition, or marketing challenges. The problem domain group (the business), therefore, must confront these issues by pursuing appropriate actions. In many cases, the business advocates the construction of innovative applications and the launch of new projects.

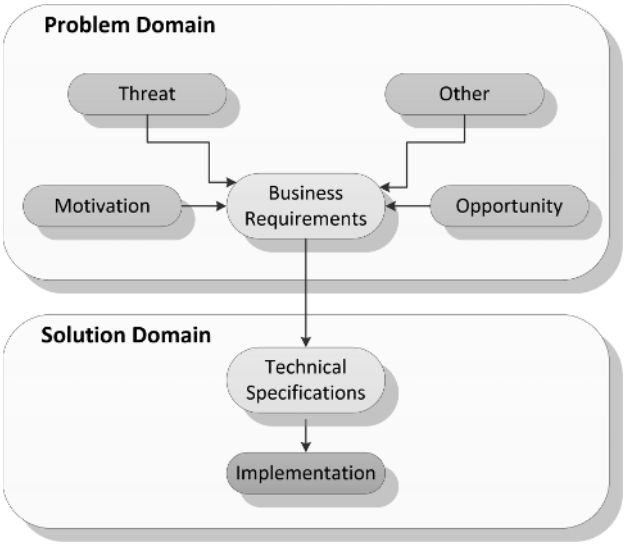

Figure 1.1: Problem and Solution Domains

So, how do business (problem domain) and IT organization (solution domain) collaborate in such cases to provide viable solutions? Figure 1.1 depicts an example of cooperation between these two domains.

The problem domain carves out business requirements to address one or more of the following issues:

Threats Internal or external threats to the business, such as security attacks and industrial espionage of trade secrets

Motivation Motivations to promote the business, such as growing market competition

Opportunities Business opportunities to gain market share, such as sponsoring new products and acquiring companies

Others Other incentives

Then the solution domain responds to the business requirements by tasking IT professionals with two chief deliverables:

- Issuing technical specifications, also known as technical requirements
- Driving technical specifications and tangible solutions, such as designing, coding, testing, deployment, and integration, to address the business problems

It must be noted that the scenario in Figure 1.1 is for demonstration purposes. It is an example that may not apply to all organizations. The business group

that issues business requirements is not always the prime mover for launching projects in the enterprise.

In other words, requirements for new projects may also be delivered by IT personnel concerned with different issues. These may include security attacks, performance degradation, insufficient computing capacity, or even technological innovation initiatives.

Recall that other divisions or departments not affiliated with the problem or solution domains may also issue requirements for a variety of projects.

Important Facts to Remember

The following list summarizes the chief takeaways of this section:

Software architects Architects are commissioned to provide software solutions, and therefore they are chiefly affiliated with the solution domain unit of an organization.

Business requirements These types of requirements are characteristically provided by problem domain professionals—in many organizations by the business department.

Technical requirements Also known as technical specifications, these types of requirements are delivered by the solution domain (IT personnel).

Project requirements issued by IT There are instances when the solution domain provides requirements for technical projects.

Other requirements Other divisions or departments within the enterprise may also be able to issue requirements for specific projects.

Scope of requirements The architect capability model requirements could support small-scale projects or large organizational initiatives.

Create a Software Architect Capability Model in Five Steps

The sections that follow elaborate on the construction process of the software architect capability model, as illustrated in Figure 1.2. The following list identifies five steps that summarize chief activities to create an efficient skill competency model:

Step 1: Provide requirements and specifications. We start creating the model from the requirements section. It includes the business requirements and technical specifications, issued by the problem and solution domain organizational entities, respectively.

Step 2: Identify software architecture practices. The next step is all about establishing the software architecture practices. They are driven by business requirements and technical specifications.

Step 3: Establish software architecture disciplines. Under each practice we then add the corresponding disciplines.

Step 4: Add software architecture deliverables. Then architecture deliverables are added to interrelated disciplines.

Step 5: Quantify skill competencies. Last, the skill competency section is created to specify the expertise level of each architect.

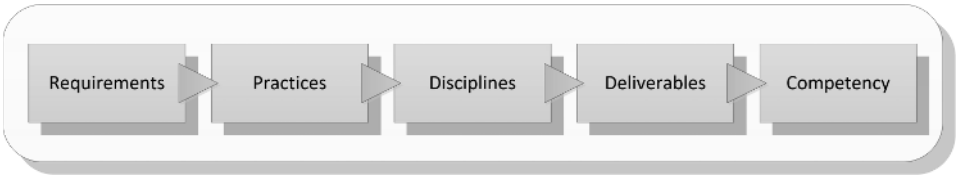

Figure 1.2: Software Architect Capability Model Creation Process

Step 1: Provide Requirements and Specifications

As indicated at the beginning of the section "Requirements Drive Architecture Deliverables," we must start with requirements to build the software architect capability model.

Before moving on, remember there is no need to be consumed by detailed requirements; they should be depicted in broad strokes. They must be abbreviated to simplify the capability model construction process. In addition, it's always a good practice to abridge the capability model rather than creating a complex and unmanageable one.

Business Requirements

As illustrated in Figure 1.3, under the problem domain section on the left side, the business requirements statement is simple and to the point. This example captures a business initiative that must be implemented: constructing a retirement planning application. Create a similar one that applies to a specific environment. It must be concise and easy to understand.

So, what might be behind the requirement statement shown in the figure? Requirement managers would probably expand on the implementation details of the retirement application. Perhaps such requirements could be laid out on umpteen pages. They may elaborate on myriad features and capabilities, such as user interfaces, instructions, retirement literature, retirement calculators, menus, pages, business rules, storage, and more.

Technical Specifications

Now, let's take a look at the solution domain, apparent on the right side of Figure 1.3. Here the technical specifications dominate the section. Again, in a software architect capability model there is no need to specify what the technical implementation details are. As shown, a short statement is enough.

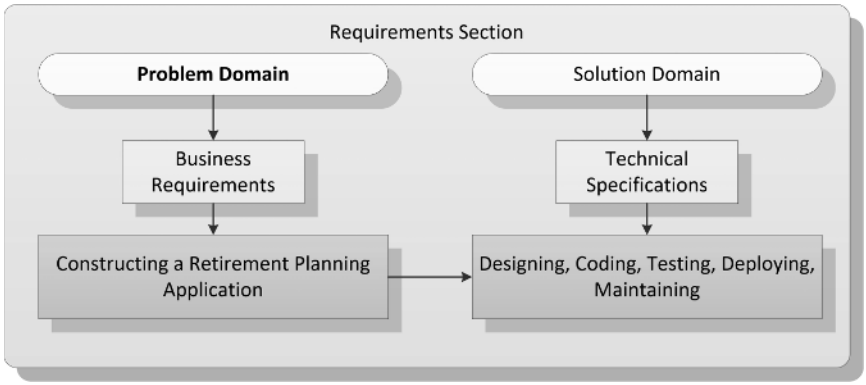

Figure 1.3: Example of Software Architect Capability Model Requirements

A longer version of the technical specifications shown might depict implementation mechanisms, technologies, and requirements for the retirement application and its commercial off-the-shelf products. Moreover, the technical specifications might include development tools and platforms, languages, scripts, storage facilities, application and integration patterns, deployment methods, configuration management, and much more.

Ensure Clear Requirements

If the software architect capability model is constructed for a project, for example, then managers are typically those who determine which talents are needed to carry out the software design duties. But such a decision, obviously, should not take place before understanding the requirements.

The peril would be that unclear requirements would most likely render confusing or ambiguous software architecture solutions.

To avoid requirements misinterpretation, in many cases, business and/or technical analysts come to aid—serving as liaisons between the organizational problem and solution domain groups. It is the duty of analysts, therefore, to construe the requirements, ensuring they are digestible and understood by managers and architects.

But if the software architect capability model is created for personal use, simply ensure that the requirements are short and easy to understand. By doing so, you eliminate the need for requirements interpretation.

Step 2: Identify Software Architecture Practices

Now that the problem and solution domains section is founded along with its issued requirements (similar to the example in Figure 1.3), it's time to establish the software architecture practice portion of the capability model.

The easiest way to understand what a *practice* is to ask random people *what* do they *do* for living. For example, doctors undoubtedly would say that they practice medicine. Correspondingly, attorneys certainly would claim that they practice law. In a similar fashion, yogis who deeply understand the science of body and mind would say delightedly that they practice yoga. And so on, and so on. So, medicine, law, and yoga are certainly different kinds of practices.

Now it's clear that a practice is something that one is professionally engaged in. It's about what a person does for a living. It's about the duties of one's livelihood. We then accordingly conclude that a software architect is an *occupation* that one pursues.

At this point comes the question that should be answered when constructing a software architect capability model: which software architecture practices are required to *meet the given requirements and technical specifications*?

Obviously, the answer is always subjective, because it depends on which software architecture practices would best provide the solutions to solve the organizational problems. And these decisions typically vary from one institution to another.

Establish Architecture Practices

Now we're ready to follow the example illustrated in Figure 1.4. It depicts two architecture practices: application architecture and data architecture. The rationale behind such a determination is based on the notion that these selected practices will offer the best solutions for the indicated requirement: constructing a retirement planning application.

This requirement insinuates that the scope of such a project will be limited to an application-level implementation—not a larger scope, such as system level. An application architect will then be the best candidate to facilitate the design and development efforts. Additionally, as discussed in Chapter 2, "Types of Software Architects," application architect tasks are confined to a narrower scope of an overall organizational solution.

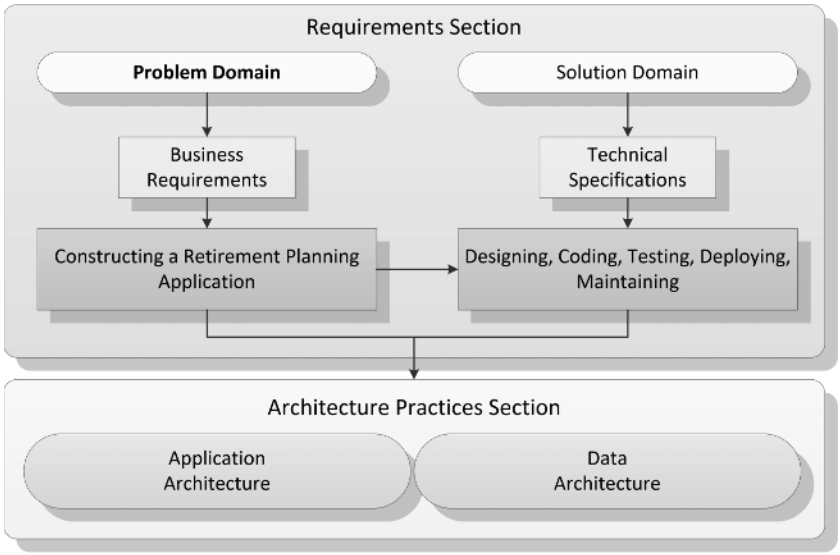

Figure 1.4: Architecture Practices

To elucidate the term *limited scope*, consider this example: a web portal that hosts retirement planning business functionality may also include reverse mortgage calculators and investment portfolio management for retirees. Consequently, the retirement planning application may be only a small part of enterprise financial offerings.

But even for such a narrow solution scope, an application architect practice calls for delivering a wide range of design artifacts. These typically include (but are not limited to) architecture blueprints—diagrams and documents depicting different application design perspectives.

By contrast, a data architect is characteristically employed to deliver different types of design artifacts. These may include data models, database schemas, table layouts, and more. Depending on how the data is designed, data architects may also insert triggers and store procedures to further manipulate the data at runtime or on different schedules.

Remember, now we're merely being asked to establish which software architecture practices will facilitate the construction of the retirement planning application. At the present time there is no need to specify the deliverables or the architecture activities required for the capability model. These will be determined later in the sections that follow.

Step 3: Establish Software Architecture Disciplines

So far, we have identified the business requirements and technical specifications. Then software architecture practices were added to the capability model.

These practices were established in our model as application architecture and data architecture. And now we are about to assign architecture disciplines for each of these practices.

So, what is a discipline? Simply put, a discipline is a *specialty*, a *field of expertise*, a subject area of knowledge. For example, a doctor who practices medicine is a subject matter expert (SME) in a branch of medical knowledge. Neurology, psychiatry, and pediatrics are only a few of the numerous fields of medical expertise. These areas of particular knowledge reveal what kinds of services a medical practitioner offers.

We therefore draw this analogy between medical and software architecture disciplines to convey that *every practitioner, in any industry, must possess proper skills to provide valuable services*. This assertion certainly applies to software architects. There is little doubt that an adroit architect would be proficient enough in providing practical solutions to certain enterprise problems.

But now there is a challenge: how would we ascertain which architecture talents, as good as they might be, can provide the best solutions for a particular project? The answer is that there should be a vetting process to ensure that an architect indeed masters the proper disciplines to facilitate a specific development initiative. Simply put, *when assessing an architect's competency, architecture disciplines then become job prerequisites*.

Apply Architecture Disciplines to Architecture Practices

Presently we are back at the same key question raised at the onset of this chapter: what do software architects do? The answer argued that not only may architects possess different skill sets, but even projects require different architecture talents. Therefore, it was impossible to provide a conclusive response. But now we've arrived at a point where we can revise this fundamental query. So, let's ask a more specific one: in what fields do architects *practice*, and what *disciplines* do they master? The sections that follow not only answer this question but also explain how to associate architecture disciplines with architecture practices.

To figure that out, let's follow the same logic demonstrated in the example in Figure 1.5. As is apparent, the business requirements and technical specifications defined earlier were driving the establishment of the application architecture and data architecture practices. Next, we're about to apply disciplines to these specific practices.

Applying Disciplines to the Application Architecture Practice

Even with the narrow solution scope of an application an architect is commissioned to deliver design artifacts prior to the development process. On one hand, an architect must fulfill business requirements and technical specifications and, on the other, adhere to design best practices. Generally, a common architect

responsibility is to break down an application into modules, components, and services. Another duty that many architects provide is to technically mentor and guide development teams.

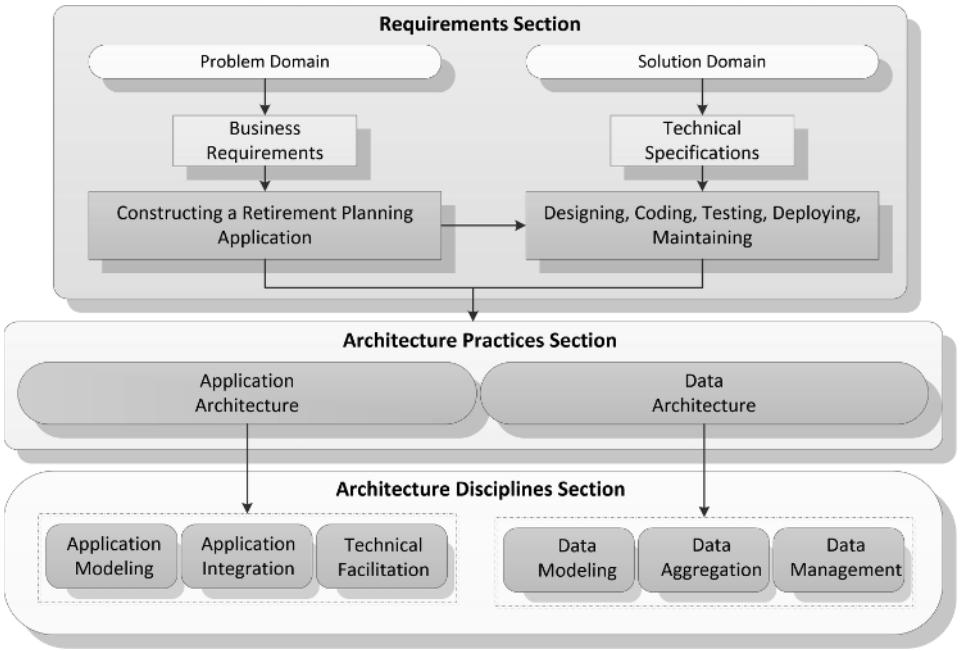

Figure 1.5: Architecture Disciplines

But in the context of the business requirements presented, the architect ought to focus only on the task at hand: constructing a retirement planning application. This example shows the three disciplines that are applied to the specific application architecture practice: application modeling, application integration, and technical facilitation.

You may wonder why the project manager opted for these particular architecture disciplines and not others. Keep in mind that every organization's software development life cycle calls for different disciplines when building products. Furthermore, it depends on how each production environment is laid out and what the integration requirements are for the specific organization.

Remember, disciplines typically vary in each organization even for the same application architecture practice.

The following list summarizes the disciplines shown in Figure 1.5 for the application architecture practice:

Application modeling This is the discipline of delivering design artifacts, visually or textually, specifying application structure, modules, components, objects, and services using an artificial modeling language.

Application integration This is the art of forming logical and physical relationships between application components, such as services and modules, to execute business functionality and technical processes.

Technical facilitation Also known as technical management, this is the duty of mentoring and guiding software development teams to deliver tangible solutions, such as applications, components, and services to be deployed to production.

Applying Disciplines for the Data Architecture Practice

In contrast, the data architecture practice is affiliated with different types of expertise—namely, architecture disciplines. These skills are leveraged to tackle a wide range of application storage requirements. Moreover, putting application transaction information into repositories is a common industry standard.

But disciplines formulated for a data architecture practice may vary from one organization to another. The same goes for projects. Not all enterprise projects require the same data architecture expertise. Therefore, when defining disciplines for a data architecture practice, stay attuned to the requirements. By the same token, follow specific organizational technical standards before adding data architecture talents to a development initiative. These standards may call for different data architecture skills.

Find in Figure 1.5 the three disciplines related to the data architecture practice. The areas of expertise in this example are required to handle the storage business requirements and technical specifications for the retirement planning application. Consider the following data architecture disciplines:

Data modeling Using their modeling skills, data architects deliver a diversity of design artifacts, such as conceptual, logical, and physical schemas. These blueprints will facilitate the construction of the retirement planning database.

Data aggregation This architecture discipline facilitates the collection of retirement rules and regulation data from distributed data sources across the organization. The retirement planning application then will be able to leverage this information for its retirement calculators.

Data management Once the retiree's data is stored in the application repository, the data management discipline becomes handy to establish a data recovery strategy, devise backup mechanisms, and even maintain data integrity.

Step 4: Add Software Architecture Deliverables

Time to work on the next piece of the puzzle: the deliverables section of the software architect capability model.

There are countless deliverables expected from each software architect during the development life cycle. It's also widely known that deliverables are expected not only at the conclusion of any enterprise initiative or project. An architect may also be chartered to provide design artifacts at a project's milestones.

So, how should these deliverables be manifested in our software architect capability model? The most logical placement for deliverables would be under the various architecture disciplines defined so far.

About Software Architecture Deliverables

But what are software architecture deliverables? Deliverables are the work of every practitioner in any field of expertise. In other words:

Deliverables are renderings of software architecture discipline activities, devised to foster organizational solutions and facilitate development goals.

For example, a software architect who is adept in the application modeling discipline would more likely be required to produce conceptual, logical, and physical design models. Similarly, a security architect who is skillful in the security modeling discipline may be asked to deliver best practices for implementing security policies.

Then who would be on the deliverables' receiving end? Remember that a deliverable must have a target audience and must facilitate stakeholders who participate in the product development life cycle. These individuals may be managers, developers, engineers, database architects, operation personnel, and more.

So, what type of deliverables should we expect from software architects? As anticipated, the answers to these questions are subjective because there is no industry-wide standard to support every single organization on the market. Companies typically develop their own sets of architecture deliverables, derived from business models, business requirements, technical imperatives, and other necessities.

Moreover, there are recognized bodies of knowledge, organizations, and publications that offer architecture life-cycle guidance and frameworks,[2] which also recommend and classify deliverables for projects. But in the context of

[2] For architecture life-cycle guidance and framework examples, refer to the Open Group Architecture Framework (TOGAF) at `www.opengroup.org/togaf` and to the Architecture Development Method (ADM) at `www.opengroup.org/public/arch/p2/p2_intro.htm`.

constructing a software architect capability model, it is crucial to adhere to deliverables requirements devised by specific organizational best practices.

If a capability model is developed for personal use, here is some all-purpose guidance for adding deliverables to the software architect capability model:

Leverage experience. Remember what the architecture deliverables were for previous projects.

Conduct research. If the personal experience is not rich enough, read articles, books, and case studies.

Consult professionals. Ask co-workers, software architects, managers, and analysts.

Add the Deliverables Section

Let's take a look at the example in Figure 1.6 to understand how the software architecture deliverables section is linked to the requirements, practices, and, finally, disciplines. To accomplish this, we simply start from the top:

1. The requirements section for the retirement planning application leads down the path to the architecture practices.
2. From there, the track downward meets the architecture disciplines section.
3. Finally, as is apparent, we find three deliverables for each of these disciplines.

The list that follows elaborates on the application architect deliverables affiliated with their three corresponding disciplines, as illustrated in Figure 1.6:

1. Application Modeling Discipline
   a. **Conceptual Architecture Model**. This is typically an application decomposition diagram depicting components and services and their relationships.
   b. **Logical Architecture Model**. An architect delivers a presentation of message flows, interfaces, and dependencies of application components, modules, and services.
   c. **Physical Architecture Model**. This pertains to diagrams illustrating the application's component deployment scheme in production, such as servers, network configuration, and routers.
2. Application Integration Discipline
   a. **Message Flow Diagrams**. These are illustrations depicting interactions and message exchanges between the application and other distributed entities in a production environment. The entities may be services, middleware, and remote data sources.

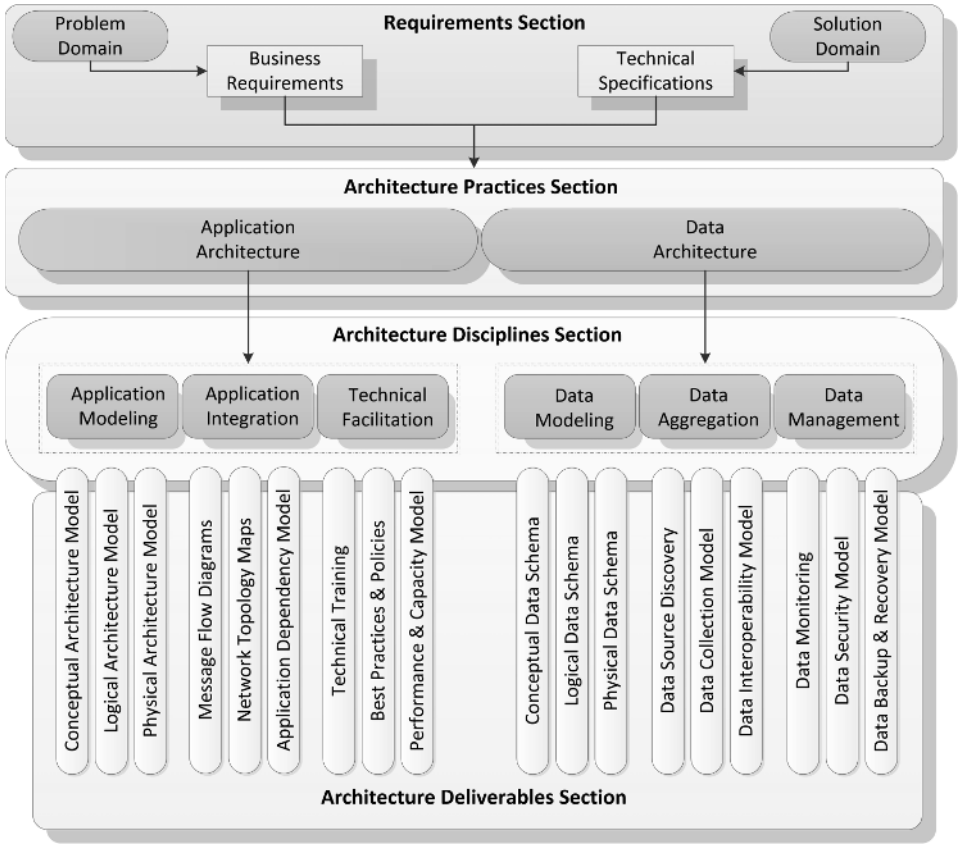

Figure 1.6: Software Architecture Deliverables Section

b. **Network Topology Maps**. These pertain to physical or logical placement of network components in a deployment environment. The logical presentation shows network connections between nodes, each of which typically represents a device. The physical topology map illustrates links between actual network devices, such as switches, routers, hubs, and modems.

c. **Application Dependency Model**. When integrating an application with other entities in production, a dependency model[3] reveals which peer applications, services, servers, and network resources it depends on. This model is also used to assess the complexity of the architecture of an application landscape.

[3] Software architects' common practice is to employ application discovery and dependency mapping (ADDM) tools to map dependencies between applications and production environment entities, such as distributed software implementations and infrastructure.

3. Technical Facilitation Discipline
    a. **Technical Training**. This provides training and guidance for a wide array of technologies utilized during the development life cycle.
    b. **Best Practices and Policies**. Every design, development, deployment, and maintenance in production efforts requires organizational best practices and policies to guide professionals with the implementation and application delivery process.
    c. **Performance and Capacity Model**. Adhering to the nonfunctional[4] requirements (NFR), the architect delivers a performance document specifying application response-time thresholds.[5] The capacity portion of this model indicates the required computing resource consumption[6] for the application.

Similarly, the corresponding data architecture's discipline deliverables, as shown in Figure 1.6, are as follows:

1. Data Modeling Discipline
    a. **Conceptual Data Schema**. The conceptual data schema represents data concepts and their associations to each other. These concepts represent chief application domains,[7] such as user, retiree, retirement plan, savings account, and retirement investment portfolio. Such abstractions will serve as building blocks for data table structures and outlines.
    b. **Logical Data Schema**. This artifact indicates the structure of information to be persisted in a data storage. The term *structure* pertains to database tables and columns, primary and foreign keys, indices, triggers, store procedures, database objects, and more.
    c. **Physical Data Schema**. This deliverable depicts the physical aspects that enable data persistence. It elaborates on how the data should be presented to the user and stored in databases. In addition, this physical perspective provides guidance for database deployment and configuration, data integration, ensuring computer capacity for data storage, and more.

---

[4] Nonfunctional requirements are specifications that describe the operational attributes and behavior of an application or system. These characteristics may include requirements such as maintainability, scalability, reusability, durability, data integrity, and fault tolerance.
[5] Performance thresholds are the maximum acceptable variances for specific metrics you can use to assess an individual project or group of projects, such as a portfolio. They are the upper-limit parameter values you can set for performance, earned value, and index calculations.
[6] *Computing resource consumption* refers to application and system utilization of deployment environment resources, such as memory, disk space, network bandwidth, and capacity of data.
[7] Application domains chiefly pertain to business applications and services.

2. Data Aggregation Discipline
    a. **Data Source Discovery.** This delivery specifies a list of data sources, third-party information providers, and repositories that the application utilizes to provide retirement planning services.
    b. **Data Collection Model**. This delivery specifies the method and when the data should be collected. The collection method pertains to network protocols, security, and interface mechanisms to ensure data integrity and acceptable transmission rates.
    c. **Data Interoperability Model**. The application architect is also chartered to ensure compatibility between the data sets collected from different data sources. This model typically specifies mechanisms for data format conversions, data filtering, and data cleansing to standardize data structures and models during transactions and message exchange.
3. Data Management Discipline
    a. **Data Monitoring**. This discipline is about employing monitoring tools to observe information exchange between the application and its environment entities, such as data repositories, consuming applications, and users. The chief reasons are to detect performance and data integrity issues and breaches to data security.
    b. **Data Security Model**. This deliverable identifies the security controls[8] that should be established to maintain data integrity and prevent internal or external attacks.
    c. **Data Backup and Recovery Model**. The application architect is responsible for guaranteeing maximum data availability. Simply put, this model elaborates on the mechanisms to replicate, store, archive, and recover data when needed.

Step 5: Quantify Skill Competencies

The last portion of the architect capability model is the skill competency section. The intention here is to evaluate the capability of an architect to deliver effectual solutions by delivering pragmatic artifacts. Put differently, this section proposes to specify the required expertise level of architects before they are recruited for any development initiative.

[8] Security control is defined by the National Institute of Standards and Technology (NIST) as "A safeguard or countermeasure prescribed for an information system or an organization designed to protect the confidentiality, integrity, and availability of its information and to meet a set of defined security requirements" (`csrc.nist.gov/glossary/term/security_control`).

In some cases, however, this skill evaluation method could also take place during an ongoing architecture engagement. The aim is to weigh in on the performance level of architects to discover the quality of their provided solutions.

This skill measurement exercise may be pursued because of these three chief reasons:

Self-assessment If the software architect capability model is constructed for personal use, the skill competency section will reveal the aptitude level of an individual to deliver effective solutions for certain projects or even wider development initiatives.

Architecture skill vetting The skill competency section will assist an organization in hiring the best possible software architecture talents for a project or any other broad scope initiative.

Assessing quality of solutions Architecture solutions should always be evaluated during and after projects or organizational initiatives. Therefore, we assert that architects' performance is directly tied to the quality of their deliverables.

Quantifying Architecture Skills

As illustrated in Figure 1.7, the capability of architects to provide effective deliverables can be measured by grading their competency on a scale from 0 to 100. Moreover, this talent scoring method can provide better architecture fitness assessment to understand the architects' ability to accomplish tasks for the corresponding architecture disciplines.

Measuring the Application Architect Skill Levels

Consider Table 1.1. It's created for identifying the skill levels of the application architect in the context of the business requirements and technical specifications. This table corresponds to the illustration in Figure 1.7 for the purpose of simplifying the information. The table demonstrates the correlation between the architecture disciplines, deliverables, and competency levels.

Table 1.1: Application Architect Competency Example

DISCIPLINE	DELIVERABLE	COMPETENCY LEVEL %
Application Modeling	Conceptual Architecture Model	100
	Logical Architecture Model	100
	Physical Architecture Model	100

DISCIPLINE	DELIVERABLE	COMPETENCY LEVEL %
Application Integration	Message Flow Diagrams	100
	Network Topology Maps	70
	Application Dependency Model	85
Technical Facilitation	Technical Training	90
	Best Practices and Policies	100
	Performance and Capacity Model	80

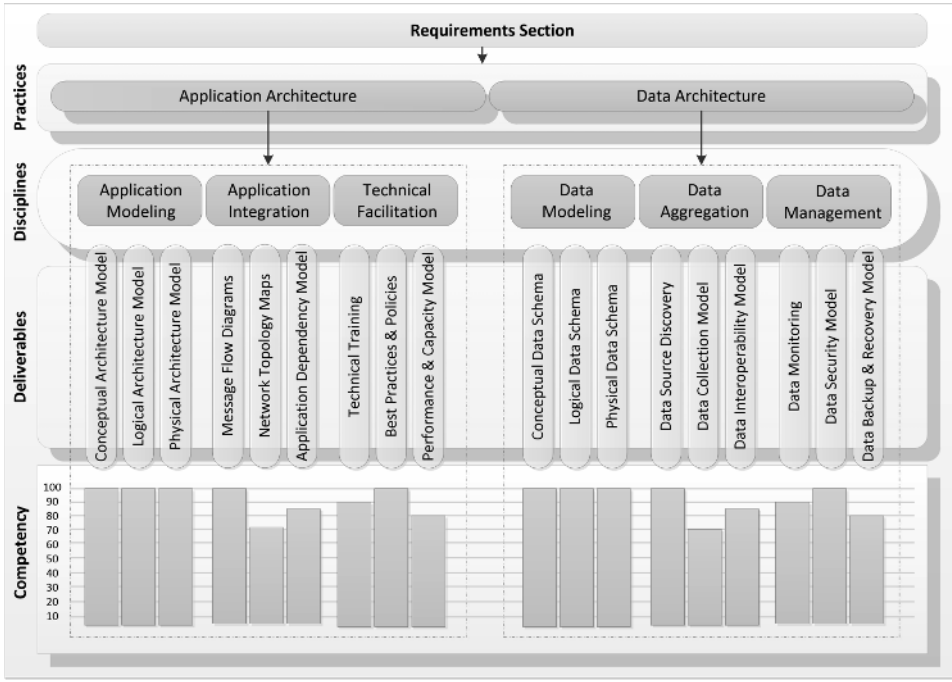

Figure 1.7: Architect Skill Competency

As shown in the table, the application modeling discipline calls for three deliverables: conceptual architecture model, logical architecture model, and physical architecture model. Note that the competency level column indicates that the skill level required for an architect to deliver these artifacts is 100 percent. This measurement could also be used for assessing the competency level of an architect to provide solutions during an ongoing project or any organizational initiative.

Furthermore, the competency levels for the application integration discipline show different scale levels for each of the corresponding deliverables, and similarly different skill competency measurements are indicated for the deliverables affiliated with the technical facilitation discipline.

Measuring Data Architect Skill Levels

In the same fashion, consider Table 1.2. It's provided here for depicting the skill level of the data architect related to the requirements and specifications. This table corresponds to the illustration in Figure 1.7. Moreover, the table demonstrates the association between the architecture disciplines, deliverables, and competency levels.

Table 1.2: Data Architect Competency Example

DISCIPLINE	DELIVERABLE	COMPETENCY LEVEL %
Data Modeling	Conceptual Data Schema	100
	Logical Data Schema	100
	Physical Data Shema	100
Data Aggregation	Data Source Discovery	100
	Data Collection Model	70
	Data Interoperability Model	85
Data Management	Data Monitoring	90
	Data Security Model	100
	Data Backup and Recovery Model	80

Consider, for example, the competency levels indicated for the data modeling discipline's deliverables:

- Conceptual data schema: 100 percent
- Logical data schema: 100 percent
- Physical data schema: 100 percent

This architect's professional capability assessment method can also be applied during an ongoing project.

Moreover, the same goes for the data aggregation discipline—note the competency levels indicated for the corresponding deliverables—and for the deliverables related to the data management discipline.

Skill Competency Patterns for Architects

The discussion about being able to assess or quantify an architect's capability to provide solutions leads to the *skill competency pattern* idea discussed in this section. In this context, therefore, the term *pattern* pertains to a *graphical representation* of an architect's ability to solve organizational challenges.

The pattern concept is not so much about the quantification of skill competency levels. It's about *visual representations* of architects' capabilities to deliver artifacts for projects. Such visual representations are typically easier to compare to each other and as a result discover gaps in knowledge and aptitude. Then later, you can draw quick conclusions about collective or individual efforts to provide solutions.

To understand better what a competency pattern is, let's take a look at the example in Figure 1.8. For the sake of simplicity, only the application architecture practice and its related disciplines, deliverables, and skill competency scale are presented.

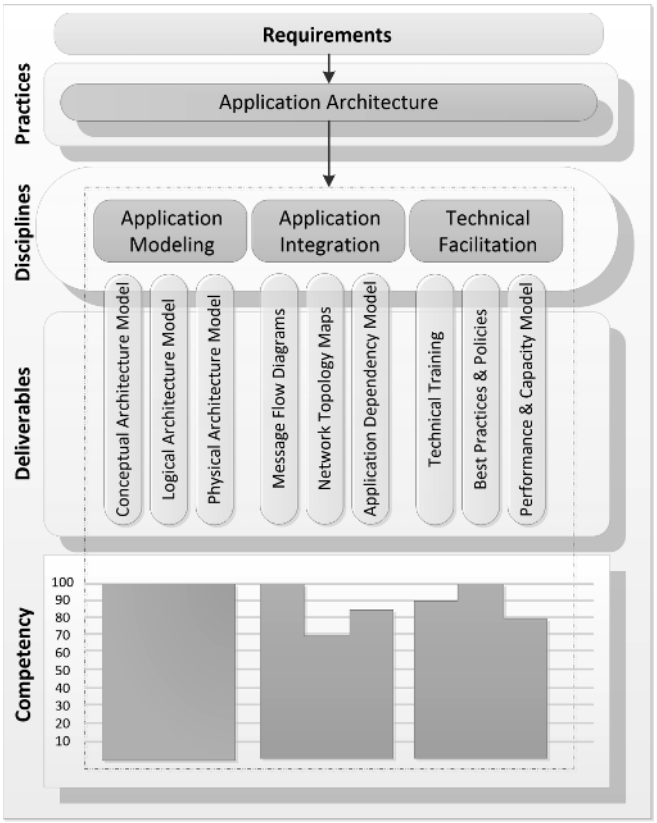

Figure 1.8: Creating a Skill Competency Pattern

As is apparent, the pattern is shown in the competency section on the bottom of this illustration. This pattern obviously signifies the architect's capability to deliver several artifacts for three software architecture disciplines: application modeling, application integration, and technical facilitation. Note that the discussed pattern still adheres to the skill competency measurements. The only difference is that the bars were united to show area patterns.

The sections that follow elaborate on the different ways individuals and organizations can utilize the competency pattern to fulfill different goals.

How Can Organizations Utilize the Skill Competency Pattern?

One of the most compelling reasons for utilizing the skill competency pattern is when architecture solutions are required for a project. Such an initiative, for example, may call for migrating legacy applications to the cloud. Apparently, the scope of such work may require a couple of architecture talents. The example in Figure 1.9 depicts the skill competency pattern of these architects: application architect 1 and application architect 2.

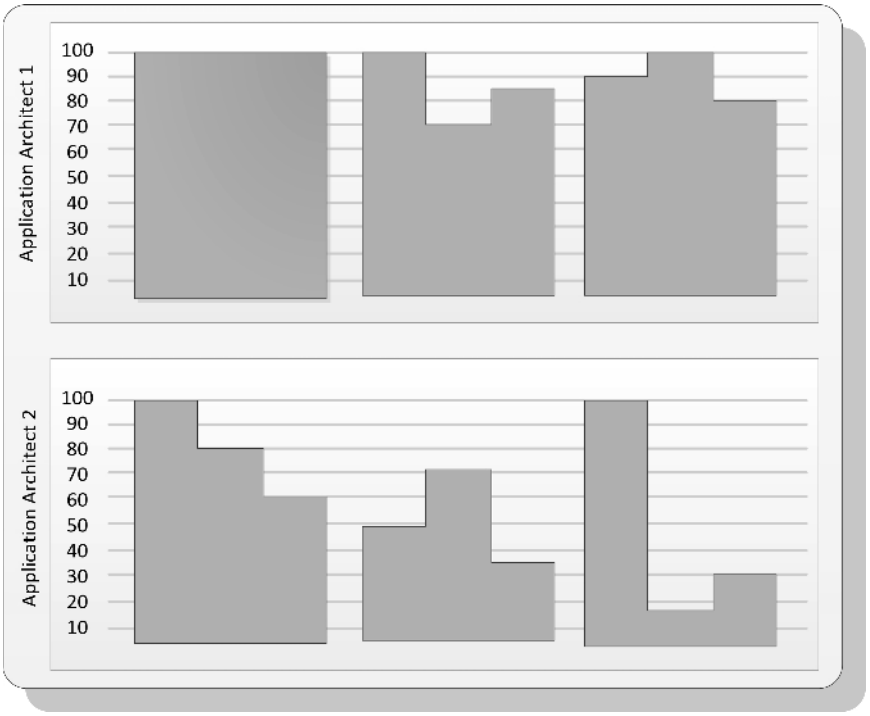

Figure 1.9: Comparing Architecture Skill Competencies

Just by visually assessing the architects' combined skill levels, it may become clear that their collective architecture capabilities might not meet the cloud migration project requirements. The skill competency pattern then can assist

management to visually assess their combined architect capability to provide solutions. Proper measures should then be taken if concerns arise. Filling in the *knowledge gap* by training or adding more architecture talents to the project will certainly boost the capability of the team to deliver effective artifacts.

The skill competency pattern, therefore, is a valuable tool for organizations to ascertain the capacity of architects to deliver effective solutions. Here is the summary of the benefits:

Discovering individual contribution Assessing and quantifying the capacity of an architect to contribute to a common enterprise effort

Comparing skill levels Comparing the competency levels of architects assigned to the same project

Revealing the collective competency gap Discovering gaps in the knowledge and capabilities of architects who participate in the same project

How an Individual Can Utilize the Skill Competency Pattern

When it comes to an individual's usage of the skill competency pattern, the promise of benefiting from it is limitless. Here are several examples:

The competency level of an ambitious and hard-working architect characteristically increases as time goes by. This proficiency progress can be reflected in the skill competency pattern created at each career milestone. In other words, an architect would be able to monitor the personal competency level progress, draw conclusions, and take proper measures to address career challenges when needed. The individual pattern would then become a personal visual representation tool to trace professional accomplishments.

An individual skill competency pattern can also be compared to other patterns that belong to co-worker architects, for example, who perform similar duties. If after such comparison gaps in skill levels are found, individuals would likely be motivated to augment their knowledge.

Moreover, it's possible to create a skill competency pattern for a job requirement if there is intimate knowledge available about the required skills. In this case, an individual skill competency pattern could be compared to the job competency pattern to reveal if there is a good match.

Consider these benefits of the individual skill competency pattern:

Tracing career progress By comparing personal skill competency patterns assessed at different career checkpoints, an individual can monitor personal growth in architecture capabilities.

Industry comparisons Individuals can assess architecture skill capabilities against common market job requirements.

Preparing for interviews It's never a bad idea to show up for an interview with an individual skill competency pattern to demonstrate strengths in particular disciplines.

Interview Questions

Interviewers will always ask questions about the role of an architect in the enterprise. These types of questions are never taken off the table. Prepare for the hardest queries and hope for the best outcomes. Therefore, the interview questions presented in the list that follows should be rigorously studied and researched until satisfactory answers are obtained. Furthermore, before an interview, use all means to acquire vast knowledge affiliated with seven fundamental questions about the architecture life cycle[9] and deliverables:

- Why do business requirements and technical specifications drive architecture deliverables?
- What is an architecture practice?
- What is an architecture discipline?
- How are architecture disciplines correlated to architecture practices?
- What are architecture deliverables?
- How should architecture deliverables be associated with architecture disciplines?
- How can architecture skills be assessed?

It's common to struggle with strategic questions, such as architecture roles and solutions. Therefore, remember to provide examples with each answer. Do not confuse an interviewer with complex or blurred definitions. State the facts clearly and keep the answers as short as possible.

And again—be prepared!

[9] The architecture life cycle depicts the activities and goals that software architects ought to fulfill during the software development life cycle. Refer to the software architecture life-cycle questions in Chapter 9, "An Outline for Software Architecture Job Interview Questions."