**2**

**Solution Architects in an Organization**

A solution architect understands the needs and goals of an organization as part of a team—all stakeholders, processes, teams, and the organization's management affect solution architect roles and their work. In this chapter, you will learn and understand the role of the solution architect and how they fit into the organization. Following that, you will learn about the various types of solution architects, and how they coexist within an organization. An organization may need a generalist solution architect, along with other specialist solution architects that are required based on a project's complexity.

This chapter will provide details on the solution architect's responsibility and how it can impact an organization's success. A solution architect wears multiple hats, and business executives heavily depend on their experience and decision making to understand the technical vision.

The solution-and-software development methodology has evolved over the last few decades, from waterfall to agile, and is needed by a solution architect. This chapter will provide details about the agile methodology and the iterative approach that a solution architect should take for the continuous improvement of solution delivery. Agile thinking is extremely important for a solution architect.

In addition to solution design, solution architects need to handle various constraints to evaluate risk and plan mitigation strategies. Quality management also plays a significant role, and should not be overlooked. The solution architect plays an essential role throughout the solution's life cycle: from requirement collection, solution design, and solution implementation, to testing, and then on to the launch.

A solution architect needs to engage regularly post-launch to ensure the scalability, ready availability, and maintainability of the solution. For broader consumer products, the solution architect also needs to work with the sales team as a technology evangelist of the product through content publishing and public speaking in various forums.

In this chapter, you will learn about the following topics:

* Types of roles for a solution architect
* Understanding a solution architect's responsibilities
* Solution architects in an agile organization

**Types of roles for a solution architect**

In the previous chapter, you learned about solution architecture and how various stakeholders impact solution strategies—now, you will understand the solution architect's role. The software solution can develop without a solution architect, depending on the project's size, but for a large project it is a requirement to have a dedicated solution architect. The success or failure of the plan depends on the solution architect.

There is always a need for someone who can make architectural decisions for the team and drive team collaboration with stakeholders. Sometimes, it is required to have multiple solution architects in the team, depending on the size of the project. The different types of solution architect are depicted in *Figure 2.1*, showing how they have different responsibilities in the organization.

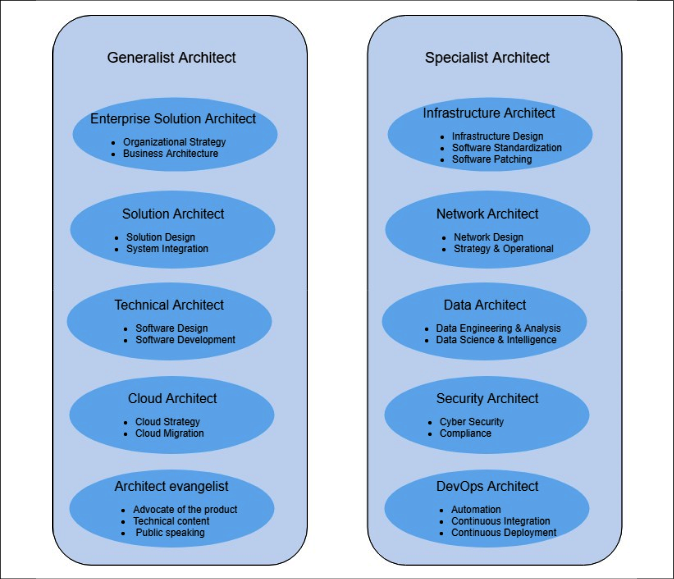


Figure 2.1: Types of solution architect

An organization can have multiple types of solution architect. Solution architects can be categorized as either **generalists** or **specialists**. Generalist solution architects have the breadth that comes from multiple technical domains, whereas specialist solution architects have very in-depth knowledge in their area of expertise, such as big data, security, or networking. A generalist solution architect needs to collaborate with a specialist solution architect to align with the project's requirements and complexity.

**Generalist solution architect roles**

The role of a solution architect varies from organization to organization and you may encounter a variety of job titles related to solution architect, the most common being generalist solution architect roles.

Details of the types of solution architect can be found in the following sections.

**Enterprise solution architect**

Do you ever think about how products launch in the information technology industry? This is where an enterprise solution role comes into the picture—they define best practices, culture, and suitable technologies. An enterprise architect works closely with stakeholders, subject matter experts, and management to identify organizational strategies for information technology and make sure that their knowledge aligns with company business rules.

Enterprise architects handle solution design across the organization; they create long-term plans and solutions with stakeholders and leadership. One of the most important aspects is to finalize which technologies should be used by the company, and ensure the company is using these technologies with consistency and integrity.

Another important aspect of the enterprise architect is defining the business architecture. In some organizations, you may see *business architect* as the job title. Business architecture fills the gap between organizational strategy and its successful execution. It helps convert a map strategy into executable action items and takes this to a tactical level for implementation.

Overall, enterprise architects are more aligned with company visions and responsibilities when it comes to defining organization-wide standards for the successful implementation of the business' vision.

**Solution architect**

In general, this book explores the role of a solution architect in a more generic way. Still, you often see solution architects with different titles, as per the organization's structure; for example, enterprise solution, software, or technical architects. In this section, you will find some distinct attributes related to the various titles. However, the responsibilities of the solution architect may overlap, depending on an organization's structure.

If you wish to know how a solution should be organized and delivered, then a solution architect plays an essential role in this context.

A solution architect designs the overall system, and how different systems integrate across different groups. A solution architect defines the expected outcome by working with business stakeholders and providing a clear understanding of the delivery objective on the part of the technical team.

The solution architect connects the dots across the organization and ensures consistency within different teams to avoid any issues that may appear late in the development process; they engage throughout the project life cycle and define monitoring and alerting mechanisms to ensure smooth operations after a product's launch. The solution architect also plays an essential role in project management by providing advice regarding resource, cost, and timeline estimation.

Overall, the solution architect gets to engage on a more tactical level compared to the enterprise architect. Sometimes, the solution architect takes on the role of an enterprise architect if more strategic involvement is required.

**Technical architect**

A technical architect may also be known as an application or software architect. A technical architect is responsible for software design and development. The technical architect works with the organization in terms of software engineering, and is more focused on defining technical details for software development by a team. They also work across the organization to understand how integration will work alongside other parts of the software module, which may be managed by other groups.

A technical architect can manage the details of API design and define API performance and scaling aspects. They make sure that the software is being developed consistently with the organization's standards and that they can easily integrate with components of other software applications.

The technical architect is a point of contact for any technical question related to the engineering team, and will have the ability to troubleshoot the system as required. For a small software development project, you may not see a technical architect role, as a senior engineer may take it up and work on software architecture design.

The technical architect mentors and supports the software engineering team by working closely with them and resolving any impediments that arise from cross-team integration or business requirements.

**Cloud architect**

The cloud architect role has only come into existence in the last decade, but as cloud adoption is increasing among enterprises, this is one role that is in great demand.

The cloud architect plans and designs the cloud environment and is responsible for deploying and managing the company's cloud computing strategies. Cloud architects provide breadth and depth for cloud services and can define the cloud-native design.

As you learned in the *Solution architecture in the public cloud* section in *Chapter 1*, *The Meaning of Solution Architecture*, use of the cloud is now very popular, and it has become the norm for organizations to move onto a public cloud. Major cloud providers, such as Amazon Web Services, Microsoft Azure, and Google Cloud Platform, are helping customers to adopt cloud platforms at exponential speed with SaaS, PaaS, and IaaS offerings. You will learn more about cloud architectures in *Chapter 5*, *Cloud Migration and Hybrid Cloud Architecture Design*.

There are a large number of enterprises that have an existing workload that they want to migrate into the cloud to utilize scalability, ease of business, and price benefits. A cloud architect can prepare a cloud migration strategy and develop a hybrid cloud architecture. A cloud architect can advise how on-premises applications will connect to the cloud and how different traditional offerings fit into a cloud environment.

For start-up businesses and enterprises starting in the cloud, a cloud architect can help to design a cloud-native architecture, which is more optimized for the cloud and uses the full capabilities it provides. The cloud-native architecture tends to be built on *pay-as-you-go* models to optimize cost and leverage the automation available in the cloud.

The cloud is now an essential part of enterprise strategy, and a cloud architect is a must-have if companies want to succeed in the modern era and keep up with the pace of innovation and automation.

**Architect evangelist**

An architect evangelist (also known as a technology evangelist) is a comparatively new role that offers a new paradigm in marketing, especially when it comes to the increasing adoption of complex solution platforms. People will always want to hear from an expert who has in-depth knowledge and the ability to answer their queries so that they can make an informed decision. Here, architect evangelists come into the picture with their expertise on a particular subject in a competitive environment.

An architect evangelist can design the architecture based on customer requirements, which resolves the customer's pain points and results in customer wins. The evangelist can be a trusted advisor for customers and partners, with a deep understanding of architectural issues, concepts, and market trends to help secure platform adoption and show revenue growth through market capture.

To increase platform adoption for the overall target audience, the architect evangelist writes public content, such as blogs, whitepapers, and articles. They speak on public platforms, whether it be an industry summit, technical talks, or conferences. They conduct technical workshops and publish tutorials to spread the word about their given product. This makes it very important for a solution architect to have excellent written and verbal communication skills; you will often see solution architects taking on technology evangelism as an additional responsibility.

**Specialist solution architect roles**

There may be other types of specialist solution architect, such as migration architect, storage architect, and machine learning architect. This, again, depends on the organization's structure. As per the project and organizational complexity, a solution architect can take on multiple roles, or different solution architects can have overlapping responsibilities.

**Infrastructure architect**

An infrastructure architect is a specialist architect role heavily focused on enterprise IT infrastructure design, security, and data center operation. They work closely with solution architects to make sure that the organization's infrastructure strategy is aligned with its overall business requirements, and they allocate appropriate resource capacity to fulfill this need by analyzing both the system requirements and the existing environment. They help reduce capital expenditure that could be utilized for operational spending to increase organizational efficiency and ROI.

The infrastructure architect is the backbone of the organization, since they define and plan overall IT resources, from storage servers to individual workspaces. The infrastructure architect creates detailed plans for procuring and setting up IT infrastructures. They define software standards, patching, and plan system updates across an organization. The infrastructure architect handles infrastructure security and makes sure all environments are protected from unwanted virus attacks. They also plan for disaster recovery and system backups to make sure business operations are always running.

In most e-commerce businesses, infrastructure architect roles become challenging, as they need to plan for periods when demands will peak, such as Thanksgiving in the USA, Boxing Day in Canada and the UK, or Diwali in India, when most consumers start shopping. They need to prepare enough server and storage capacity to accommodate the peak season, whose workload may be ten times higher than normal, thus increasing the cost of IT infrastructure. A system will be sitting idle for most of the year, outside of the peak season.

They need to plan for cost optimization and better user experience, which is another reason they may use the cloud to fulfill additional capacity and scale on-demand to reduce the cost. They need to ensure that systems are occupied while supporting the growth of new expertise.

Overall, an infrastructure architect needs to have a good understanding of data center operation and the components involved, such as heating, cooling, security, racking and stacking, server, storage, backup, software installation and patching, load balancers, and virtualization.

**Network architect**

Have you ever wondered how giant enterprises with multiple locations for offices or stores are connected? Here, the network architect comes into the picture, as they orchestrate an organization's network communication strategy and establish communication between IT resources, giving life to the IT infrastructure.

A network architect is responsible for designing the computer network, **Local Area Network** (**LAN**), **Wide Area Network** (**WAN**), internet, intranet, and other communication systems. They manage organizational information and network systems, and ensure low network latency and high network performance is available for users to increase their productivity. They establish secure connectivity between user workspaces and the internal network using **Virtual Private Network** (**VPN**) connectivity.

The network architect works closely with the infrastructure architect; sometimes you see this as an overlapping role to ensure all IT infrastructures are connected. They work with the security team and design the organization's firewall to protect against unethical attacks. They are responsible for monitoring and protecting the network via packet monitoring, port scanning, and putting an **Intrusion Detection System** (**IDS**) and **Intrusion Prevention System** (**IPS**) into place. You will learn more about IDS/IPS systems in *Chapter 8*, *Security Considerations*.

Overall, a network architect needs to have a good understanding of network strategies, network operations, secure connections using VPN, firewall configuration, network topology, load balance configuration, DNS routing, and IT infrastructure connectivity.

**Data architect**

Any solution design revolves around data, and it is mostly about storing, updating, and accessing it regardless of whether it is focused on customers or products. In the last decade, data growth has risen exponentially—not long ago, gigabytes of data were considered as big data, but now, even 100 terabytes of data are deemed to be normal—you can even get a 1-terabyte computer hard disk.

Traditionally, data was stored in a structured relational way. Now, most data is in an unstructured format generated from resources such as social media, the **Internet of Things** (**IoT**), and application logs. There is a need to store, process, and analyze data to get useful insights, which is where the data architect role comes into the picture.

The data architect defines a set of rules, policies, standards, and models that govern the type of data that's used and collected in the organization database; they design, create, and manage the data architecture in an organization. A data architect develops data models and data lake designs to capture the business's **key performance indicators** (**KPIs**), and enable data transformation. They ensure consistent data performance and data quality across the organization.

The primary customers for a data architect are as follows:

* Business executives using **Business Intelligence** (**BI**) tools for data visualization
* Business analysts using a data warehouse for greater data insight
* Data engineers performing data wrangling using **Extract, Transform, and Load** (**ETL**) jobs
* Data scientists for machine learning
* Development teams for application data management

To fulfill organizational needs, a data architect is responsible for the following:

* Selection of database technology
* Structured and unstructured data store choice
* Streaming and batch data processing
* A data lake as the centralized datastore
* A relational database schema for application development
* Data warehousing for data analysis and BI tools
* Datamart design
* Data security and encryption
* Data compliance

You will learn more about data architectures in *Chapter 13*, *Data Engineering for Solution Architecture*. Overall, the data architect needs to be aware of different database technologies, BI tools, data security, and encryption to make the right selection. As machine learning is becoming more prominent among enterprises, the emergence of dedicated machine learning architect roles is expected.

**Machine learning architect**

As we know, **artificial intelligence** (**AI**) and **machine learning** (**ML**) have been a hot topic for quite some time now, and more companies are pivoting to implement ML in their enterprise solutions stacks. The public cloud has accelerated organizations' adoption of ML, with easily accessible infrastructure and tools. ML helps to tackle customer problems in many ways, including to develop personalization, provide forecasts, and detect fraud. In addition to that, ML can solve many daily challenges for IT leaders, software architects, and solution architects, such as automation of security, infrastructure, disaster recovery, and monitoring of solutions. This has contributed to the rise of ML architect roles, which carry the following responsibilities at a very high level:

* Applying systems thinking to implement/adopt ML in the enterprise software stack
* Identifying and analyzing tools for ML and AI implementation
* Architecting the information/data architecture for ML
* Modifying the current software stack and tools to make way for ML integration
* Operationalizing ML with continuous monitoring and improvements

An ML architect creates an AI/ML solution architecture by applying architecture best practices while considering the performance optimization, security, compliance, reliability, cost optimization, and operational excellence of the AI/ML solution design. It's not the case that every problem can be solved with AI/ML, and ML architects should understand how ML solutions fit in an agile enterprise environment. They must put together AI/ML architecture design by providing a detailed understanding of design pillars, advanced design patterns, antipatterns, and the cloud-native aspect of modern AI/ML tech stack design. You will learn more about ML in *Chapter 14, Machine Learning Architecture*.

**Security architect**

Security should be the top priority of any organization; there are multiple instances when large and well-established organizations have gone out of business due to a security breach. Organizations not only lose customer trust but also experience legal complications due to security incidents. There are various industry compliance certifications, such as **Organizational Security** (**SOC2**), **Finance Data** (**PCI**), and **HealthCare data** (**HIPAA**), that are in place to ensure organization and customer data security, which a company needs to adhere to, depending on the nature of their application.

Looking at the critical nature of security, organizations need to research and design the most robust security architecture for their projects, and that's where a security architect is necessary. A security architect works closely with all teams within the organization and external vendors to make sure security is a high priority. A security architect's responsibilities include the following:

* Designing and deploying the implementation of the network and computer security in the organization
* Understanding the company's technology and information systems, and safeguarding the security of the computers in the organization
* Working with a variety of settings, such as securing company networks and websites
* Planning vulnerability testing, risk analysis, and security audits
* Reviewing and approving the installation of a firewall, VPN, and router, and scanning the server
* Testing final security processes and making sure they work as expected
* Providing technical guidance to the security teams
* Making sure applications comply with industry standards as required
* Making sure data is secure with the required accessibility and encryption

Security architects are expected to understand, design, and guide all aspects of security related to data, network, infrastructure, and applications with a variety of tools and techniques. You will learn more about security and compliance in *Chapter 8*, *Security Considerations*.

**DevOps architect**

As a system gets more complex, there is more potential for human error, which can lead to an additional effort being required, increased cost, and even reducing quality. Automation is the best way to avoid failure and improve overall system efficiency. Automation is not optional—if you want to be agile and move faster, automation is a must.

Automation can be applied anywhere, whether it is testing and deploying applications, spinning up infrastructure, or even ensuring security. Automation plays a critical role, and a DevOps architect role is to automate everything, wherever possible. DevOps is a combination of practices and tools that assist in delivering an application at a faster pace.

It allows the organization to serve its customers better and stay ahead of the competition.

In DevOps, the development team and operational team work together in parallel. For a software application, a DevOps architect defines **continuous integration and continuous deployment** (**CI**/**CD**). In CI, automated builds and test runs happen before the development team merges its code changes into a central repository. CD expands upon CI by deploying all code changes to a production environment after the build and test stage.

The DevOps architect automates infrastructure deployment, known as **Infrastructure as Code**, which is highly prevalent in the cloud environment. DevOps can utilize tools such as **Chef** and **Puppet** for instructed automation, or use cloud-native tools if the workload is in a cloud environment. They may choose to automate infrastructure using script such as **Ansible** and **Terraform**. Infrastructure automation provides excellent flexibility for the development team in the aid of experimentation, and enables the operations team to create replica environments.

For smooth operation, a DevOps architect plans monitoring and alerting with automated communication in the event of issues or any significant changes. Any security incidents, deployment failures, or infrastructure failures can be monitored automatically, and alerts can be sent to the respective mobile devices or email accounts of a team when required.

The DevOps architect also plans for disaster recovery through different deployment methods. Organizational **Recovery Point Objective** (**RPO**) is the volume of data loss that an organization can tolerate. **Recovery Time Objective** (**RTO**) suggests how much time the application can take to recover and start functioning again. You will learn more about DevOps in *Chapter 12*, *DevOps and Solution Architecture Framework*.

**Understanding a solution architect's responsibilities**

Now we have broken down the various roles of a solution architect, we will next take a look at the details of the responsibilities of a solution architect. A solution architect is a technical leader in a customer-facing role, which comes with many responsibilities. The primary responsibility of a solution architect is to convert organization business visions into a technical solution and work as a liaison between businesses and technical stakeholders. A solution architect uses broad technological expertise and business experience to ensure the success of the solution's delivery.

A solution architect's responsibilities may differ slightly based on the nature of the organization. Often, in a consulting organization, a solution architect may be dedicated to a particular project and customer, while in a product-based organization, a solution architect may be working with multiple customers to educate them on a product and review their solution design.

A solution architect carries various responsibilities at different stages of the application development cycle, even before the project is kicked off. During the project incubation phase, the solution architect works with business stakeholders to prepare and evaluate the **request for response** (**RFR**) document. Once a project is kicked off, the solution architect analyzes the requirements to decide on the feasibility of technical implementation, while defining non-functional requirements such as scalability, high availability, performance, and security. A solution architect understands various project constraints and makes the technology selection by developing a proof of concept. Once development starts, the solution architect mentors the development team and adjusts both technical and business needs. After the application has launched, the solution architect makes sure that the application performs as per the defined non-functional requirements, and identifies the next iteration based on user feedback. You will learn more about the solution architect role at various stages of the product development life cycle in this section. Overall, a solution architect holds the following primary responsibilities detailed in *Figure 2.2*.

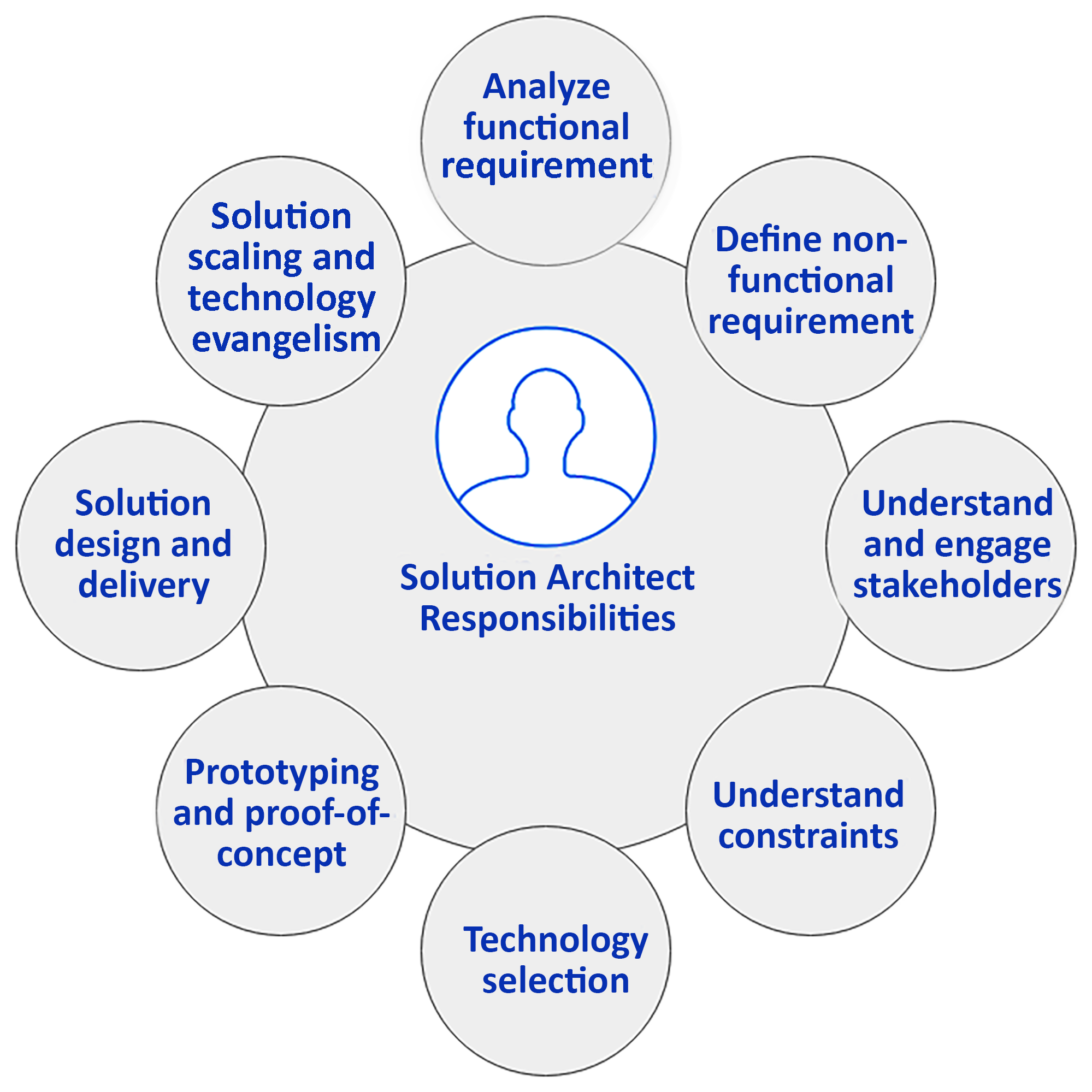


Figure 2.2: Solution architect's responsibility model

As shown, there are various significant responsibilities for a solution architect. In the upcoming sections, you will learn about the various aspects of the solution architect's responsibilities.

**Analyzing user requirements**

Business requirements are at the center of any solution design, and they are defined in raw terms when a project starts. It is necessary to engage a diverse set of groups from the beginning, which includes the technical capability to identify requirements.

The business stakeholder defines requirements, and multiple adjustments are warranted when it comes to the technological evolution of the project. To save effort, it is necessary to engage solution architects while defining the user requirement documentation.

The solution architect designs the application, which may impact the overall business outcome. This makes requirement analysis a critical skill that a solution architect should possess. A good solution architect needs to have the skills of a business analyst and the ability to work with a diverse set of stakeholders.

Solution architects bring a broad range of business experience with them—they are not only technical experts but also have a good knowledge of the business domain. They work closely with the product manager and other business stakeholders to understand all the aspects of requirements. A good solution architect helps the product team uncover hidden requirements, which a non-technical stakeholder may not have thought about from an overall solution perspective.

**Defining non-functional requirements**

**Non-functional requirements** (**NFRs**) may not be visible to users and customers directly, but their absence may impact the overall user experience in a negative way, and hamper the business. NFRs include critical aspects of the system, such as performance, latency, scalability, high availability, and disaster recovery. The most common NFRs are shown in *Figure 2.3*.

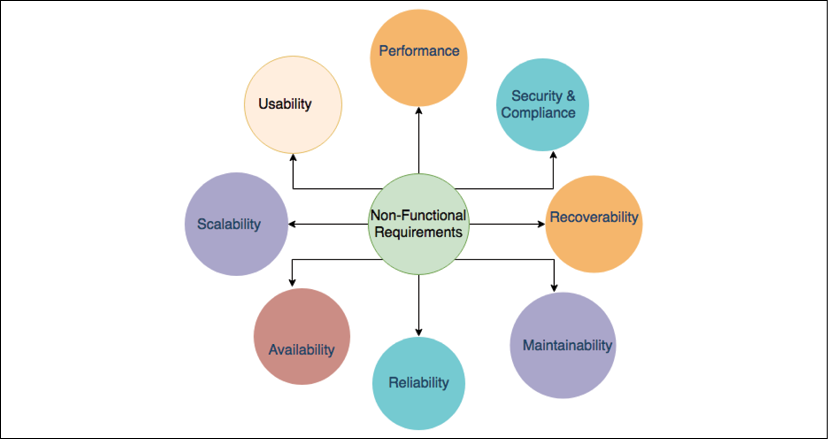


Figure 2.3: NFRs in a solution design

Consider the following NFRs:

* **Performance**:
  + What will the application load time be for users?
  + How can we handle network latency?
* **Security and compliance**:
  + How can we secure an application from unauthorized access,
  + Protect an application from malicious attacks,
  + And adhere to local laws and audit requirements?
* **Recoverability**:
  + How can we recover an application from an outage,
  + And minimize recovery time in the event of an outage?
  + How can we recover lost data?
* **Maintainability**:
  + How can we ensure application monitoring and alerts?
  + How can we ensure application support?
* **Reliability**:
  + How can we make sure the application performs consistently,
  + And inspect and correct glitches?
* **Availability**:
  + How can we ensure the high availability of an application,
  + And make an application fault-tolerant?
* **Scalability**:
  + How can we meet the increasing demand for resources?
  + How can we achieve a good scale for a sudden spike in utilization?
* **Usability**:
  + How can we simplify an application's use,
  + Achieve a seamless user experience,
  + And make the application accessible to a diverse set of users?

Depending on the nature of the project, however, there may be certain NFRs that are suitable only for that particular project (for example, voice clarity for a call center solution).

You will learn more about these attributes in *Chapter 3*, *Attributes of the Solution Architecture*.

The solution architect becomes engaged in a project from a very early stage, which means they need to design a solution by gauging requirements across the stakeholders within an organization. The solution architect needs to ensure consistency in solution design across system components and requirements. The solution architect is responsible for defining NFRs across groups and different components since they make sure that the desired usability of a solution is achieved across the board.

NFRs are an integral and essential aspect of solution design, which tends to slip when teams are too focused on business requirements, which can impact the user experience. A good solution architect has the primary responsibility of conveying the importance of NFRs and ensuring that they are implemented as part of solution delivery.

**Engaging and working with stakeholders**

A stakeholder is anyone who has an interest in the project, whether directly or indirectly. As well as the customer and user, it may also be the development team, sales, marketing, infrastructure, network, support team, or the project funding group. Stakeholders can also be internal or external to the project. Internal stakeholders include the project team, sponsors, employees, and senior management; external stakeholders include customers, suppliers, vendors, partners, shareholders, auditors, and the acting government of a country.

Stakeholders often have a different understanding of the same business problem as per the context they find themselves in; for example, a developer may look at a business requirement from a coding perspective, while an auditor may look at it from one of compliance and security. A solution architect needs to work with all technical and non-technical stakeholders.

Solution architects possess excellent communication skills and negotiation techniques, which help them to ascertain the optimal path for a solution while keeping everyone on board. A solution architect works as a liaison between technical and non-technical resources and fills the communication gap. Often, those communication gaps between a businessperson and the technical team become a reason for failure. The businessperson tries to look at things from more of a feature and functionality perspective, while the development team strives to build a more technically compatible solution, which may sometimes lean toward the non-functional side of the project.

The solution architect needs to make sure both teams are on the same page, and that the suggested features are also technically compatible. They mentor and guide the technical team as required and put their perspective into simple language that everyone can understand.

**Handling various architecture constraints**

Architecture constraints are one of the most challenging attributes of solution design. A solution architect needs to manage architectural constraints carefully and be able to negotiate between them to find an optimal solution. Often, these constraints depend on each other, and emphasizing one limitation can inflate others. The most common constraints are presented in *Figure 2.4*.

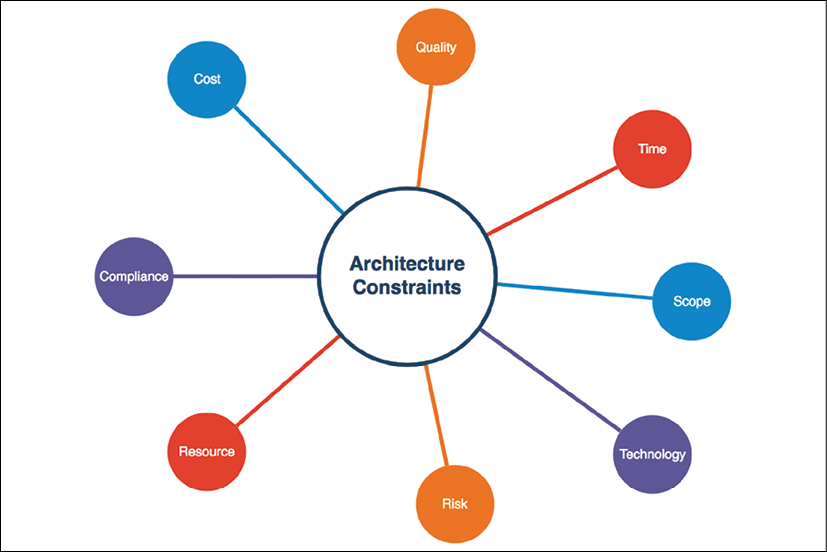


Figure 2.4: Architectural constraints in a solution design

As shown, solution design helps us understand the following attributes of an application:

* **Cost**:
  + How much funding is available for solution implementation?
  + What is the expected ROI?
* **Quality**:
  + How closely should outcomes match functional and non-functional requirements?
  + How can we ensure and track the quality of the solution?
* **Time**:
  + When should the output be delivered?
  + Is there any flexibility regarding delivery time?
* **Scope**:
  + What is the exact expectation from business and customer requirements?
  + How does the requirement gap need to be handled and accommodated?
* **Technology**:
  + What technology can be utilized?
  + What flexibility does using legacy versus new technologies provide?
  + Should we build in-house, or source from a vendor?
* **Risk**:
  + What can go wrong, and how can we mitigate it?
  + What is the risk tolerance of stakeholders?
* **Resource**:
  + What is required to complete solution delivery?
  + Who will work on the solution's implementation?
* **Compliance**:
  + What are the local legal requirements that might impact the solution?
  + What are the audit and certification requirements?

There could be more specific constraints related to a project, such as the way in which data is stored in a country due to government regulation, and opting for in-house development due to security concerns. Handling constraints can be very tricky.

A solution architect needs to balance constraints and analyze the trade-off of each; for example, saving costs by reducing resources may impact the delivery timeline.

Achieving a schedule with limited resources may affect quality, which in turn increases cost due to unwanted bug fixes. So, finding the balance between cost, quality, time, and scope is significant. **Scope creep** is one of the most challenging situations that a solution architect may be faced with, as it can negatively impact all other constraints and increase the risks of solution delivery.

It is essential for a solution architect to understand all the aspects of every constraint and to be able to identify any resulting risk. They must put risk mitigation plans into place and find a balance between them. Handling any scope creep can help a lot in delivering the project on time.

**Making technology selections**

Technology selection is the key aspect of a solution architect's role, and may involve the most complexity. There is a broad range of technologies available, and a solution architect is required to identify the right ones for the solution. The solution architect needs to have breadth and depth in their knowledge of technologies to make the best decision, since the chosen technology stack can impact the overall delivery of the product.

Each problem can have multiple solutions and an available range of technologies. To make the right selection, a solution architect needs to keep functional requirements and NFRs in mind, and define selection criteria while making a technology decision. The selected technology needs to consider different perspectives, whether the goal is the ability to integrate with other frameworks and APIs, or to meet performance requirements and security needs.

A solution architect should be able to choose the technology that not only satisfies current requirements but also scales for future needs.

**Developing a proof of concept and a prototype**

Creating a prototype is probably the most fun part of being a solution architect. To choose a proven technology, a solution architect needs to develop a **proof of concept** (**POC**) in various technology stacks to analyze their fit for the functional and non-functional requirements of the solution. The solution design POC is when a solution architect is trying to figure out the building blocks of the solution.

The idea of developing a POC is to evaluate technology with a subset of critical functional implementations, which can help us to decide on a technology stack based on their capabilities. It has a short life cycle and is limited to being reviewed by experts within a team or organization.

After evaluating multiple platforms using a POC, the solution architect may proceed with prototyping to a technology stack. A prototype is developed for demonstration purposes and given to the customer so that it can be used to secure funding. POCs and prototyping are far from being production-ready; solution architect builds have limited functionality, which can prove to be a challenging aspect of solution development.

**Designing solutions and staying through delivery**

Solution architects work on solution design after understanding different aspects of functional requirements, NFRs, solution constraints, and technology selection. In an agile environment, this is an iterative approach where the requirements may change over time and need to accommodate the solution design.

The solution architect needs to design a future-proof solution, which should have strong building blocks and be flexible enough to adjust to changes that can occur due to user demands or technology enhancements. For example, if the user demands increase ten times, then an application should be able to scale and accommodate user demands without significant changes to the architecture. Similarly, if new technology, such as ML or blockchain, gets introduced to solve a problem, your architecture should be able to accommodate them; for example, using AI to build a recommendation system on top of existing data for an e-commerce application.

However, the solution architect needs to be careful about drastic changes to the requirements and apply a risk mitigation plan. For future-proof design, you can take the example of a loosely coupled microservice architecture based on RESTful APIs. These architectures can be extended to new requirements and have the ability to integrate easily. You will learn more about different architecture designs in *Chapter 6*, *Solution Architecture Design Patterns*.

*Figure 4.5* contains a flowchart that shows the solution delivery life cycle. The solution architect is involved in all the phases of solution design and delivery.

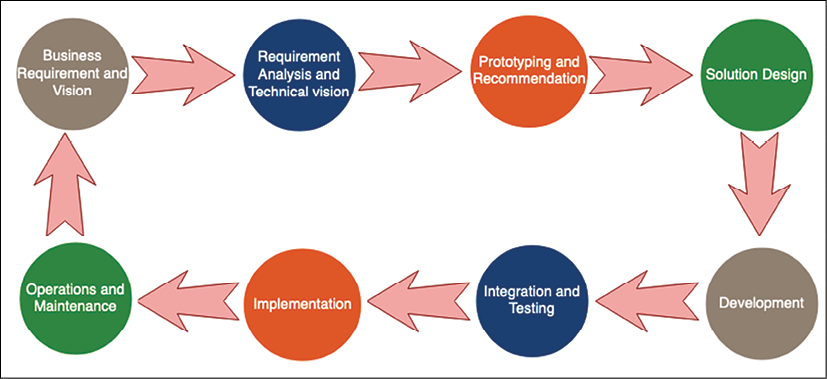


Figure 2.5: Solution delivery life cycle

As shown, the solution delivery life cycle includes the following, with the ways in which a solution architect is involved:

* **Business Requirement and Vision**: A solution architect works with business stakeholders to understand their vision.
* **Requirement Analysis and Technical Vision**: Analysis of the requirements, defining a technical vision in order to execute the business strategy.
* **Prototyping and Recommendation**: Makes a technology selection by developing POC and showcasing prototypes.
* **Solution Design**: A solution architect develops solution designs in line with an organization's standards and in collaboration with other impacted groups.
* **Development**: Works with the development team on solution development, and as a bridge between the business and technical team.
* **Integration and Testing**: Makes sure that the final solution is working as expected with all functional and non-functional requirements.
* **Implementation**: Works with the development and deployment team for smooth implementation and guides them through any issues.
* **Operation and Maintenance**: Ensures logging and monitoring are in place and guides the team on scaling and disaster recovery as required.

The overall life cycle is an iterative process. Once the application goes into production and customers start using it, more requirements may be discovered from customer feedback, which will drive the product vision for future enhancements.

The solution architect has major ownership during solution design, in which they do the following:

* Document solution standards
* Define high-level design
* Define cross-system integration
* Define different solution phases
* Define an implementation approach
* Define a monitoring and alert approach
* Document the pros and cons of design choices
* Document audit and compliance requirements

Solution architects are not only responsible for solution design: they also help project managers with resource and cost estimation, defining the project's timeline and milestones, the project's release, and its support plan. The solution architect works through different phases of the solution life cycle, from design to delivery and launch. The solution architect helps the development team overcome obstacles and hurdles by providing expertise and a broad understanding.

**Ensuring post-launch operability and maintenance**

The solution architect plays an integral role after the solution's launch with respect to product operability. To handle the increasing user base and product utilization, a solution architect should know how to scale the product to meet demands and ensure high availability without impacting the user experience.

In unforeseen events such as outages, a solution architecture guides infrastructure, IT support, and software deployment teams to execute a disaster recovery plan for business process continuation. The solution architect satisfies the organization's **Recovery Point Objectives** (**RPOs**) and **Recovery Time Objectives** (**RTOs**). RPOs define how much data loss an organization can tolerate in terms of the volume of data lost during the outage interval—for example, a loss of 15 minutes of data. RTOs define how much time the system should take to get back up and running. You will learn more about RTOs and RPOs in *Chapter 12*, *DevOps and Solution Architecture Framework*.

In the event of performance issues due to an increase in demand, the solution architect helps scale the system horizontally to mitigate application bottlenecks, or vertically to alleviate database bottlenecks. You will learn more about different scaling mechanisms and self-healing in *Chapter 9*, *Architectural Reliability Considerations*.

The solution architect plans to accommodate any new requirements in an existing product that arise from usage patterns or for any other reason. They can make changes to NFRs based on monitoring user behavior; for example, users may leave a page if it takes more than 3 seconds to load. The solution architect works through this and guides the team in handling issues that may occur post-release.

**Working as a technology evangelist**

Being an evangelist is the most exciting part of the solution architect role. The solution architect increases product and platform adoption by spreading the word through public forums. They write blogs about solution implementation and conduct workshops to showcase the potential benefits and the use of technology platforms.

They build mass support for technologies and help establish a standard. A solution architect should be passionate about technology. They should be an excellent public speaker and possess excellent writing skills to perform the technology evangelist role.

**Solution architects in an agile organization**

In the last half decade, you may have seen the rapid adoption of the **agile methodology**. In this competitive market, an organization needs to be proactive to rapid changes, and bring output to the customer very quickly. Fast innovation and release can only be possible if organizations are adapting quickly and respond to change faster to accommodate user demand, which means there must be flexibility built into every part of the organization and solution architecture.

To be successful in an agile environment, a solution architect needs an agile mindset, and must adopt a rapid delivery method by continuously working with stakeholders to fulfill their needs. First, let's understand a little bit more about the agile methodology. This is a vast topic, and in this section, we will see a high-level overview.

**Why the agile methodology?**

Agile can create and respond to changes to make a profit in a fast-moving business environment. In today's competitive environment, where technology is moving fast (which results in a high probability of changes and customer demand), agile is the answer to coping with this situation and gaining a competitive edge.

Nowadays, all successful organizations are customer-driven: they take frequent feedback from end users on their products and use that feedback to expand their user base. Agile helps gather feedback from users to continuously adapt it to new software releases, and most of the time everything has a high priority. To deal with this situation, you need agile.

Executive management provides funding and seeks transparency. They demand productive output to increase ROI, and a solution architect wants to win their confidence by showing incremental development of the product. To create transparency for a project and keep track of its budget and delivery timeline, you need agile. When you continuously want to engage your stakeholders by showing them a demonstration of the product, and when development and testing are part of the same cycle, you need agile.

The preceding scenarios are situations where the agile methodology is required to keep the organization ahead with robust delivery and customer feedback.

Agile is *able to quickly move* in a timebox manner, which means you timebox activities in a short cycle and take an iterative approach for product development, instead of working on the entire product to develop and deliver it in one go. The agile methodology advocates seeking continuous feedback by keeping customers and stakeholders engaged closely, involving them in every phase of product development, adapting feedback into requirements, evaluating market trends, and working with them to prioritize the stakeholders. Then, the development team take up the prioritized requirements, conduct technical analyses, design, develop, test, and deliver.

Everyone works as a unified team toward one goal and breaks the silo mindset, with agile thinking helping the technical team to understand the requirements from the customer's perspective and respond to changes quickly and efficiently. This is the reason why most companies want to go agile. The agile methodology is fast and easy to adopt using many tools that are available on the market, such as **JIRA**, **VersionOne**, and **Rally**. You may face some initial challenges while developing agile thinking, but the benefits very much outweigh any challenges that an organization may face when moving toward adopting the agile methodology.

**Agile manifesto**

Applying any form of agile requires a clear understanding of the four values stated in the agile manifesto. Let's understand these values:

* **Individuals and interactions over processes and tools**: Processes and tools always help complete the project. Project stakeholders, who are a part of the project, know how to implement the plan and how to deliver the successful result with the help of tools for project delivery. But the primary responsibility for project delivery is the *people* and their collaboration.
* **Working software over comprehensive documentation**: Documentation is always an essential process for any product's development. In the past, many teams only worked to collect and create a repository for documents, such as high-level design, low-level design, and design change, which later help to achieve qualitative and quantitative descriptions of the product.
  + With the agile methodology, you focus on the deliverable. Therefore, according to this manifesto, you need documentation. However, you also need to define how much documentation is vital to the continuous delivery of the product. Primarily, the team should focus on delivering software incrementally throughout the product's life cycle.
* **Customer collaboration over contract negotiation**: Earlier, when organizations worked on a fixed bid or time and material projects, the customer was always involved in the first and the last stages of the software life cycle—they were outsiders who were not involved in product development. By the time they finally got a chance to see the product after launch, the market trends had changed, and they lost the market.
  + Agile believes that customers share equal responsibility for the product's launch, and that they should be involved in every step of development. They are part of the demonstration, giving feedback based on new market trends or consumer demand. Since the business is now part of the development cycle, these changes can be attained by being agile and having continuous customer collaboration.
* **Responding to change when following a plan**: In the current fast-paced market in which customers demands change with new market trends, businesses also keep on changing. It is vital to make sure there is a balance between frequently changing the requirements and welcoming the changes with agile, since sprint cycles vary from one to three weeks. Responding to change means that should anything change in the specification, then the development team will accept the change and show the deliverable in sprint demonstrations to keep winning the confidence of the customers. This manifesto helps the team understand the value of *welcoming changes*.

The **agile manifesto** is a tool that is used to establish basic guidelines for adopting an agile methodology. These values are the core of all agile techniques. Let us understand the agile process in more detail.

**Agile process and terminology**

Let's get familiar with the most common agile terms and how they bind together. Here, you will learn about the agile **scrum** process, which is widely adopted. The agile scrum process has a small sprint cycle of one to three weeks, depending on the project's stability, but the most common is a two-week **sprint cycle**, which can also be called a development cycle.

These sprints are development cycles where the team will analyze, develop, test, and deliver a working feature. The team takes an iterative approach and creates a working building block of the product as the project progresses with each sprint. Each requirement is written as a user story that keeps a customer persona in mind, and makes the requirement clearly visible.

The agile scrum team has varied roles. Let's understand the most common ones, and how the solution architect collaborates with them:

* **Scrum Team**: This consists of the product owner, Scrum Master, and development team. Analysts, technical architects, software engineers, software testers, and deployment engineers are part of the development team.
* **Scrum Master**: A person in this role facilitates all scrum ceremonies (which you will learn about in the next section), keeps the team motivated, and removes impediments for the team. The Scrum Master works with the solution architect to remove any technical blockers and get technical clarification for business requirements.
* **Product owner**: This is a businessperson who is a customer advocate. The product owner understands market trends and can define priorities within the business. The solution architect works with the product owner to understand the business' vision and keep it aligned with the technical view.
* **Development team**: Product implementation is carried out, with the team also being responsible for the project's delivery. They are a cross-functional team that is committed to continuous and incremental delivery. The solution architect needs to work closely with the development team for smooth product implementation and delivery.

**Scrum ceremonies**

The sprint cycle includes multiple activities that are performed to manage development, which are often called scrum ceremonies. Those scrum ceremonies are as follows:

* **Backlog grooming**: Grooming is a timebox meeting in which the product owner, solution architect, and business meet to discuss backlog stories, prioritize them, and create a consensus for sprint deliverables.
* **Sprint planning**: In sprint planning, the Scrum Master facilitates groomed stories that are assigned to the scrum team based on the team's capacity.
* **Sprint Daily Standup**: The Daily Standup is a very efficient way of collaborating, where all team members meet in one place and discuss their latest day's workload, what plans they have for the day, and whether they are facing any problems. This meeting is meant to be short and straightforward, at around 15 minutes in length. The Standup is the platform that the solution architect uses to collaborate with the development team.
* **Sprint demonstration**: During demonstrations, all stakeholders gather and review the team's work of what they have done in a sprint. Based on this, the stakeholder accepts and rejects the user stories. The solution architect makes sure that the functional and non-functional requirements have been met. During this meeting, teams collect feedback from the product owners and solution architect and look at what changes were made.
* **Sprint retrospect**: The retrospect is conducted at the end of each sprint cycle and is where the team inspects and adopts best practices. The team identifies things that went well, along with what they should continue to improve, as well as things that they can do better in the next sprint. Sprint retrospect helps the organization apply continuous improvements while working on their delivery.

**Agile tools and terms**

Let's learn about some agile tools that help drive team metrics and project progress:

* **Planning poker**: Planning poker is one of the most popular estimation techniques in the agile methodology, where the Scrum Master conducts a planning poker session to estimate user stories when a sprint starts. During this activity, each user story will be evaluated based on its complexity. Team members use comparative analysis to give story points for each user story, which helps the team understand how much effort is required to complete the user stories.
* **Burndown chart**: A burndown chart is used to monitor sprint progress and to help the team understand how much work is pending. The Scrum Master and the team always follow the burndown chart to make sure there is no risk in the sprint, and reuse that information to improve the estimation next time.
* **Product backlog**: The product backlog contains a collection of requirements in the form of user stories and epics. The product owner continuously updates the backlog and prioritizes requirements during sprint grooming. An epic is a high-level requirement, and product owners write a user story to refine them. The development team breaks down these user stories into a task, which is an executable action item.
* **Sprint board**: The sprint board contains a collection of user stories listed for the active sprint. The sprint board provides transparency, as anyone can look at the project's progress for that particular sprint cycle. The team refers to the board on a daily standup to determine overall work progress and remove any obstructions.
* **Definition of Done**: This means all user stories should pass the *Done* criteria that have been set up by the solution architect and product owner in collaboration with stakeholders. Some of these criteria are as follows:
  + The code must be peer reviewed
  + The code should be unit tested
  + Enough documentation has been generated to explain code flow and API design
  + Code quality is to an acceptable standard as defined by the team and organization
  + Code writing is to an acceptable standard as defined by the team and organization

**Agile versus waterfall**

Waterfall is one of the oldest and most traditional software development methodologies that organizations used to follow. In this section, you will learn about the difference between waterfall and agile and why organizations need to move over to agile. We are not going to look at the details of the waterfall process; instead, we will point out the key differences:

* Agile methodologies help change mindsets from the traditional method to an agile mindset. The motivation for this is to move from the waterfall method to agile methods in order to achieve maximum business value and win customer confidence.

This makes agile an advocate for customer collaboration at each step, while also providing transparency. The waterfall method tends to be more project-and document-centric, where customers were involved at the end phase.

* The waterfall method is more helpful for the project when all requirements are unambiguous and the sequence of their deliverables is also known, which helps remove any unpredictability as requirements are very straightforward. The agile methodology is helpful for companies that want to keep up with the market trend and have increased pressure from the customer. They need early releases for their products and have to be adaptive to changes in the requirements.
* Agile projects are delivered in a small iterative manner with the highest quality and to achieve business value. Many agile teams work in parallel across the sprint to provide a shippable solution for the product at the end of every sprint cycle. As every sprint has a small deliverable and keeps building on top of previous releases, the customer continuously gets to see a working model of the product. Waterfall has a long cycle, and stakeholders get to see the final product at the end, which means there isn't much scope left to accommodate changes.
* The agile process ensures the team is progressing toward its goals, and that the project will be completed on time by putting checkpoints in at every sprint cycle. In traditional waterfall methods, there is no frequent checkpoint that can ensure that the team is on the right path and verify whether the project will be completed on time, which may cause ambiguity.
* In the agile methodology, the customer always collaborates with the product owner and the team. This collaboration ensures that they observe and review the small, shippable product. Agile also ensures that work is being done, enabling progression to be shown to stakeholders. However, in the waterfall method, there is no such customer interaction until the project ends.

Agile is the most adaptive methodology, since fast-moving technologies and businesses are becoming unpredictable and need high team productivity. Agile supports inspecting and adapting cycles, which creates a balance between demand and control.

**Agile architecture**

What comes to mind when thinking about the solution architect in an agile model? There are many myths, such as thinking that the solution architecture is a very complex activity, and with agile you will be asked to submit your design right away or in the next sprint cycle. Another myth is that the agile architecture will not be robust to such architecture design and development, or that testing is not possible.

A solution architect in an agile environment needs to follow an iterative re-architect concept by inspecting and adapting the approach. It's about choosing the right solution for enterprises, communicating well, taking continuous feedback, and modeling in an agile way. The development team needs a solid foundation and the ability to adapt to changing requirements; they need guidance and mentoring from a solution architect.

The foundation of the agile architecture should be reducing the cost of changes, reducing unnecessary requirements by challenging them, and creating a framework to reverse incorrect requirements rapidly. The agile architect builds prototypes to minimize risk and plans for change by understanding them. They design the prototype while balancing the needs of all stakeholders and creating a loosely coupled architecture that can easily integrate with other modules.

Agile architecture advocates designing decoupled and extendable interfaces, automation, rapid deployment, and monitoring. Solution architects can build decoupled designs using microservice architecture and rapid deployment using test framework automation with a continuous deployment pipeline. You will learn more about various loosely coupled architecture patterns in *Chapter 6*, *Solution Architecture Design Patterns*.

**Summary**

In this chapter, you have learned how the solution architect fits into an organization, and how different kinds of solution architect roles may coexist within one. There are generalist solution architect roles, such as the enterprise solution architect, solution architect, technical architect, cloud architect, and architect evangelist. The generalist solution architect has a broad knowledge of technology and may develop in-depth expertise in a particular area. The specialist solution architect dives deep into other required areas of the project. The specialist solution architect possesses in-depth knowledge of their area of expertise, with some of the most common specialist solution architect roles being the network architect, data architect, security architect, infrastructure architect, and DevOps architect.

You have also seen solution architect responsibilities in great detail. Solution architects wear multiple hats; they work with stakeholders across the organization, analyze functional requirements, and define non-functional requirements. The solution architect ensures consistency and standards across the organization, and provides technology recommendations and solution prototypes. The solution architect handles various project constraints, such as cost, quality, scope, and resources, and finds a balance between them.

The solution architect helps the project manager to estimate costs and the resources required and to define a timeline while being present throughout the project from design to launch. During the project's implementation, the solution architect makes sure that the stakeholder's expectations have been met and works as a liaison between the technical and business teams. The solution architect engages in post-launch application monitoring, alerts, security, disaster recovery, and scaling.

The benefits of the agile methodology have also been covered. We took a brief overview of this, exploring the roles, tools, and terminology, and how agile differs from the traditional waterfall method. You learned about the traits of agile architecture and how solution architects should make their architecture more flexible and agile.

In the next chapter, you will learn about the different attributes of the solution architecture that you should consider while designing a solution. These attributes include architecture security, scalability, availability, reliability, fault tolerance, extensibility, portability, interoperability, operational excellence, performance efficiency, cost optimization, and self-healing.