System Architectures
An Overview

IT 4 Recruiters

IT4Recruiters.com
Suite 300 #288
115 Penn Warren Dr
Brentwood, TN 37027

Contact Name: Michael Meloche

Overview

One could argue that architectures are a foundation that businesses are built on. As in construction, a solid architecture will mean a building that will stand the test of time, be very functional, and be capable of handling additions without structural failure. Were no architecture is used, buildings tend to be constructed poorly, have nonfunctioning floor plan designs, areas structurally unsound, and have a tendency to collapse.

Just like construction, architectures for developing Systems and Software are the blue prints or road maps used to determine the correct building blocks for building organized systems. Without architectures businesses would be left with chaotic masses that are unmanageable, unsustainable, and threaten the very existence of an organization. As we go through the different types of architectures, try to the keep the construction example in mind. As you will soon discover, the more complex a System Architecture can become the more it resembles the construction of a building. Each type of architecture has its own set of blocks. These can be integrated with other existing sets of blocks or architectures to build larger complex systems.

This document will cover a variety of topics covering the common areas of System Architectures. The technical details of System Architectures will be glossed over to limit how technical this document gets, while still trying to provide a level of knowledge that a typical recruiter or hiring manager might find useful.

What are System Architectures?

System Architectures define, build, and implement the workings of a complex system. This is done with the use of different models, structures, behaviors, and systems to build an organizations desired system. These systems can include; types of desktop machines, email clients, network solutions, and servers used with in an organization.

Why are they important?

A good System Architecture ensures systems are implemented that analyze the business needs, behaviors, relationships, and structural integrity. This will ensure that the system implemented will be stable, reliable, and grow with the organization. A poorly implemented or missing architecture can lead to staggering costs, potential security threats, and growth problems.

Who uses them?

In today's market most small to large organizations use some form of System Architecture. There are many examples of System Architectures all around. Email, online games, and web applications are primary examples of software and hardware that are built using System Architectures.

Keywords

Here are some useful definitions to keep in mind while reading this document:

- Client - can be low-end machines or software that communicates with a sever across a network connection.

- Server - tends to be a more expensive mainframe or high-end computer that handles all the processing requests from multiple client computers. In many cases, the server acts as an application, email, telephony, database, or repository server in some cases...all of the above.

- API (Application Program Interface) - is a set of programming instructions and standards used by companies to allow external programs or web applications to interact with their services. For example a company who wants to track auction information on EBay could use Ebay's api to interact with Ebay's auction services.

- Form (Web Form) - a web page (i.e. login, registration, etc...) designed for user input that is sent over the internet to a server for processing.

- Threads - Software threads are processes managed by an OS. While Hardware threads are processes managed by the processor.

System Architectures Snapshot

Standalone

A system or program that is self contained and can run on a device or OS without the need for any additional services, database, software, or network connections. It does not play well with other services or applications and is considered a one stop shop. Also, a system that is designed for one specific purpose in mind. Fully self contained and therefore, its considered to be one and done.

Whats does that mean, it means that by itself it is designed for one specific purpose. The standalone architecture can be installed or a runnable service, it can also be hardware or software. However, it is its own, self contained system or service. Designed so that it does not rely on other pieces of hardware or other services nor does it require network access.

Standalone applications have been around since the early days of software development. When creating standalone applications for a single device or OS this approach can benefit a company by getting the application to the market quickly with only a few developers with the required development skills, and a small support model.

On the downside most standalone application are designed for a specific device or OS making them hard to scale. This tends to cause issues when the device or OS does not match the exact application requirements. This can cause the application to perform in an unpredictable and sometimes fatal way. Porting the application to other devices and/or Operating Systems can be very expensive. Generally requires additional developers with varied development skills and a much larger support model. This has lead to the decline of this model in the past few decades to be replaced by newer, faster, and more scalable architectures.

Standalone:
A system or program that is self contained and can run on a device or OS without the need for any additional services, database, software, or network connections.

Designed so that it does not rely on other pieces of hardware or other services nor does it require network access.

Standalone applications have been around since the early days of software development.

When a application is only needed to run a single system with no other dependencies like networks.
Such as:
- Notepad for Windows

Problems with Stand-Alone architecture:
- Hard to Scale
- Expensive to support
- Tend to be OS specific
- Not very popular anymore

Client-Server

In general, client-server is perceived as sharing processes between a client and server machine typically with multiple clients sharing a server. Most people have probably been exposed to client-server in some form or fashion without ever knowing it. For example, most companies use some form of security to make their users log into a computer. During the process of logging into the computer the service involved connects to the remote machine (usually a domain server). This machine sits on the company network and handles the security requests for user login information. When the actually login begins, the client machine communicates with the server to validate the login process. Once that login is verified the server sends back a message informing the client if the login is valid or not. Once logged into the computer or laptop other services become available. These services can also model the client server. For example connecting to a network folder is a connection to other backend servers or machines on the network. You can also try to run services or applications that are on other machines through the company network.

Here is an image depicting a client-server model architecture:

Types of Client-Servers

Thin client - an service or machine that does little of the processing. The majority of the processes are off loaded to services running on remote servers or appliances, such as data warehouses. A simple example of a thin client would be a web browser with a simple form for collecting information. All the client does is capture the user inputs to the questions. Then the user clicks the send button then client passes all the collected

Client Server:
Perceived as sharing processes between a client and server machine typically with multiple clients sharing a server.

Typical uses for a Client-Servers are:
- Email
- Authentication
- Data Storage

Types of Client-Servers:
- Thin Client
- Fat Client
- Rich Client
- Peer-to-Peer

Thin Clients:
Service or machine that does little of the processing.

A typical use for Thin-Client is data entry or information gathering from a user.

One of the largest uses for thin clients today is with mainframe computers.

information thru the network, internet, or local server to handle all the processing, data manipulation, business requirements for the request.

Fat client (Thick client) - Fat client on the other hand is the reverse. Most of the processing and servicing of a fat client are handled on the client machine. For example, all the business logic on a website can be handled on the client machine instead of that information going back to a specific server. The client handles all the manipulation and processing for whatever application or service your trying to do. A really good example of this would be using some form of accounting software or payroll systems where a lot of the payroll information is all processed on the fronted and all your doing is storing information to the backend for some payroll systems.

Rich Clients- a mix of thin/thick that will have a combination of services running locally and remotely to help balance load. So if you think in terms of web applications and web browser, they have a lot of information on the screen for a form. There are a lot of clicks and actions that can be done on the screen. Most of the process is done within the application locally on the machine. This includes displaying the information, manipulating data, and calculations when an action is clicked. The information is sent to the backend server where additional work of about equal value and load is done. This information can also be sent off to other message brokers, agencies, or databases. When the server is finished the information is communicated back to the client.

Fat Clients:
Most of the processing and servicing are handled on the client machine

A typical use for a Fat-Client would be accounting software where a lot of the information is processed on the client machine.

Rich Clients:
a mix of thin/thick that will have a combination of services running locally and remotely to help balance load.

Most web applications use Rich-Client in their implementations.

Peer-To-Peer:
The ability to group a network of services together to allow for a service to be shared among the members without a central server

Most commonly recognized for its ability to allow multiple systems to share files simultaneously over a network like Napster and BitTorrent.

Additional communication back forth between the two may be required to keep the data in sync.

Peer-to-Peer - The ability to group a network of services together to allow for a service to be shared among the members without a central server. This removes the need for a costly file server. Eliminates potential network bottlenecks of a file server by sharing the service load across a group of computers. In the entertainment world a lot of people have heard of Napster, BitTorrent, and file sharing systems. These are systems that were setup to allow individual machines to host files and allow other people to download those files spread out across a network of peer machines. Everyone has a particular file on their machine. For example you have a music file "A" on a network of 3 machines. Machine 4 wants to get a copy of music file "A". Machine 4 would put out a request to download music file "A". Music file "A" would be downloaded form one of the other machines based on availability. This is a form of peer to peer.

Tier Architecture:
breaks down systems into individual layers and separates the service by application and hardware.

3-tier can be done on a single machine; however, it is typically done on three.

The 3-Tier layers are:
- Presentation
- Business
- Data

Presentation Layer:
Typically on a client machine or browser which focuses on the graphical interface presentation for capture user entered data.

Business Layer:
This is typically where the data is processed and additional business logic is handled.

Data Layer:
Handles the storing of the data into a file server or database.

N-Tier / 3-Tier

A Tiered architecture breaks down systems into individual layers and separates the service by application and hardware. Each layer becomes a tier and can be separated further into smaller services. These additional services can run locally or remotely to improve performance and scalability.

- 3-tier can be done on a single machine; however, it is typically done on three. The focus is separating services into three distinct layers of presentation, business, and data. This enables these services to be executed in multiple threads.

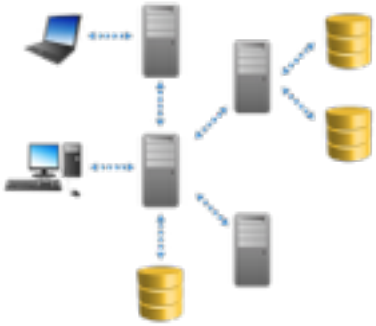
Here is a visual example depicting a 3-Tier architecture:

- N-tier, also known as "multi-tier", expands the 3-tier architecture by breaking it down further into additional layers. These layers can have multiple presentation, business, data, physical, network, transport, session, and application layers.

Distributive Architecture

This approach takes N-tier to the next level. Distributive breaks the services into even smaller components and then spreads these components out across machines and networks. Communicating is done by passing services, messages, and requests back and forth to synchronize their actions. Some widely used examples of these systems are online massively multiplayer online games (MMOs) such as "World of Warcraft" and the Amazon's system of servers.

Here is a visual example depicting a distributive architecture:



N-tier:
Also known as "multi-tier", expands the 3-tier architecture by breaking it down further into additional layers.

Distributive:
Each layer can be broken down into smaller and smaller pieces to spend the work out across multiple machines and networks.

One examples of these systems is Amazon's system of servers.

Cloud:
Is a combination of many architecture styles wrapped into one.

If you have an internet connection, you have access to "The Cloud".

Rise in popularity has been brought on by the rise in use of smart phones, tv's, and tablets

Cloud

Today we find ourselves surrounded by mobile devices of many kinds. From smart phones, tv's, and tablets there is a common architecture that is fueling these trends. Cloud-oriented architecture or "The Cloud", is a combination of many architecture styles wrapped into one. This means we can now incorporate many different services and have the ability to access, store, and manipulate data through the internet. This gives systems the freedom to communicate without being serviced from one physical device or network. If you have an internet connection, you have access to "The Cloud".

A common cloud component is Platform as a Service (PAAS). All information and hardware related to the service is hosted by a provider and offered to users as a service. Using this model the user does not need to install any special hardware or software to run the application.

Cloud's are typically Service-oriented, made up of API's, or Web Services. More organizations are reviewing their current service practices in favor of a more cloud friendly approach. These approaches can lead to huge cost savings and and can be a revenue generator.

By developing their systems using the cloud, organizations distribute their services to the masses. Users no longer have to go to the store and buy a piece of software, or worry about missing updates that could potentially put their system at risk. All the end-user needs is a browser to connect to the software.

Here is a visual example depicting a cloud architecture:

Common Cloud Components:

- Software as a Service (SAAS)
- Infrastructure as a Service (IAAS)
- Platform as a Service (PAAS)

Cloud's are typically Service-oriented, made up of API's, or Web Services.

Microsoft 365 and Adobe Cloud already have adopted these models allowing users to stay up to date on the latest tools.

Candidates with a degree tend to get paid more than those with only work experience.

Market Skill Set

When a system architect is just starting out, companies will take candidates right out of college with a formal degree. Developers without a degree that have actual experience working with architectures will also be accepted. Do not be surprised if an organization looks to pay less for a candidate with equivalent experience rather than a degree. It all depends on what the organization is looking for and what expertise the candidate already has.

Companies looking to hire system architects generally list them with some variation of these words; computer, engineering, system, technology, architects, and others from the topics in this document. These positions tend to be permanent rather than contract. Be sure to clarify with the hiring manager exactly what they want in this role and who they will be working with.

System Architect Types	
Tier I (0-5 yrs exp)	Early candidates may have basic understanding of one or two principles but no real work experience. Most knowledge will be obtained from classes, training, or on the job experience thru projects. These positions tend to be more development in nature.
Tier II (5-10 yrs exp)	Experience working with multiple architectures on projects and a stronger ability to apply the architecture principles. At this stage the candidate will have stronger communication skills working with managers and business staff to identify and build architectures.
Tier III (10-20+ yrs exp)	Senior skilled position with experiences implementing multiple architectures. Strong communication skills with teams and departments to identify and maintain systems architecture.

Certifications

System Architecture certifications are rare. There may be some certifications that do provide some level of testing or certification relating to System Architectures and are applicable.

Companies looking to hire system architects generally list them with some variation of these words:

- computer
- engineering
- system
- technology,
- architects

Architect roles are often broken into "tiers" but this is not universal.

Common mistake companies make when looking a System Architect:

- Overload requirements
- Improper compensation for tier
- Want a software Architect instead

System Architecture certifications are rare.

Finding the right architect starts with understanding what the company is really looking for in the position.

System Architectures Conversations

Architects serve a vital role in the health and growth of a companies infrastructure. It is important to understand that a System Architects role is crucial and should not be taken lightly nor overlooked when companies start to build their business or products. Poor architectures can lead to higher costs and over taxed systems which prevent a company from performing. It is the role of the system architect to work within the departments of a company to identify the best approach.

Clarifying Questions

Whether you are trying to fill a position for a system architect or you see things in the job description like client-server, N-Tier, Cloud, etc.. Here are some questions to help find resources that are good candidates to fill a position:

- What System Architecture, if any is being used?
- What do you mean when you say you want a system architect?
- How important are System Architectures to the role being filled?
- Is this strictly an System Architect position or will this candidate be required to have other skills as well?
- Does the organization have mentors for the System Architecture? are they hoping to find mentors?
- Will the candidate need to know multiple System Architectures or just some of the them? Which ones?
- How extensive is the current System Architecture? Are they just starting to build one? Are they looking to replace the existing architecture?
- Do they need someone who understands the existing architecture to maintain it and continue to grow it?
- Do they need to know client server?
- What parts of client server do they need to know?

Questions to help identify what Architect position the company needs:

- What System Architecture, if any is being used?
- What do you mean when you say you want a system architect?
- How important are System Architectures to the role being filled?
- Is this strictly an System Architect position?
- Is this a mentored position or will they be mentoring someone?
- Will the candidate need to know multiple System Architectures or just some of the them?
- How extensive is the current System Architecture?
- Do they need someone who understands the existing architecture?
- Do they need to know client server?

Crossover/Complementary Skills

A general rule with System Architectures is, if a candidate has knowledge or experience with a higher level of architecture, then they should be able to understand or perform a job in a System Architecture that in a step down. For example, a candidate with knowledge in client-server should have no problem doing standalone. Also, candidates with knowledge in cloud should be able to do any of the architectures listed in this paper. What a candidate can or cannot do may require additional training. It is not typically difficult for a candidate to go up one step in the architecture hierarchy. For example, going from client server to N-tier. Going up multiple steps is not advisable and probably not what the client has in mind for their ideal candidate.

Depending on the position there may be other needs for development, documentation, and testing. Candidates with areas in web development, mobile development, system documentation, or testing could be a bonus and help a candidate with weaker System Architecture skills. When in doubt, talk to the hiring manager and get examples of additional skill sets that may apply outside of the job description.

For example, a hiring manager looking for a client server architect might be interested in someone who has experience developing web applications or using an application server like Weblogic, Tomcat, or Websphere. Another manager may be in need of someone with additional skills with network security or Active Directory experience to manage their security policies.

Weasels

System Architect weasels generally fall into a couple of categories. They either try to convince the customer that they have full System Architecture experience when they do not, or they try to show experience they do not have in a specific System Architecture. In both cases weasels tend to use terms and buzz words they do not understand to sell how knowledgeable they are. This makes it fairly easy to spot the weasels by asking them to explain the buzzwords they use. A weasel will often avoid specifics in their explanation/definition (or might be completely wrong) where a truly experienced candidate will provide a specific definition and be able to provide examples from their experience.

Many candidates have some knowledge of System Architectures and will try and slip the buzz words into their resume to give the

Typically candidates can only advance up one position of System Architecture at a time.

For example:
Going from position doing Stand-Alone Architecture to a Client-Server Architecture position is acceptable, but going from Stand-alone to Cloud can not easily be done.

Candidates with knowledge in a higher level of Systems Architecture like Cloud, should have no problem stepping into any of the Architectures listed in this paper.

2 common weasel types:

1. try to convince the customer that they have full System Architecture experience when they do not.
2. try to show experience they do not have in a specific System Architecture

impression that they have skills as an architect. Being an expert requires a little more understanding than just a base definition answer. When pressed as to why they used a particular architecture these candidates will not be able answer. Others will give completely incorrect answers due to their misconceptions or lack of knowledge with the specific architecture. A true Systems Architect will quickly jump into working examples of existing projects with a detailed description and definition of how they work. Spend some time using the areas discussed in this document as a guideline to Google the web for additional information to prepare yourself when speaking to the hiring manager to determine which skills they need and how they are used within the organization.

To verify a claim of skills ask for practical examples and prior experience. This will quickly weed out the weasels.

Screening Questions

Here are some questions that you, the recruiter, can ask the candidates to help clarify if the candidates meet the requirements for these positions. Incidentally, you can also see if candidates have other areas of expertise overlooked or not on their resume. If these can help the candidate apply for the position, then they should be included on the resume. This will help the candidate show their skills and qualities for this particular position.

Good screening questions are typically open ended and intended to lead to the candidate giving long form answers rather than a simple, or canned answer.

- What are the characteristics of client-server architecture? This can be modified for a specific architecture. Response varies based on architecture so you may need to ask for specific details of the architecture to determine whether the answer is correct.
- What is the difference between thin client and thick client?
- Explain your current project's System Architecture? This will help determine the kind of experience the candidate has and help identify if they truly understand System Architecture or their understanding is that of a software architecture.
- What are the business benefits of client-server architecture? This can be modified for a specific architecture. Response varies based on architecture so you may need to ask for specific business benefits of the architecture to determine whether the answer is correct.
- Is there a particular System Architecture you prefer or are most comfortable with? Why? This is a more advanced question, but may be very important when specific architecture is required or preferred.

Great Interview Questions:

1. What are the characteristics of client-server architecture?
2. What is the difference between thin client and thick client?
3. Explain your current project's System Architecture?
4. What are the business benefits of client-server architecture?
5. Is there a particular System Architecture you prefer or are most comfortable with?

System Architecture Resources

Resources for more information

For readers that want to know more about System Architectures including some specifics, here are some great resources:

Source Materials

architecture. (n.d.). Dictionary.com Unabridged. Retrieved May 21, 2015, from Dictionary.com website: http://dictionary.reference.com/browse/architecture

http://upload.wikimedia.org/wikipedia/commons/thumb/5/51/Overview_of_a_three-tier_application_vectorVersion.svg/800px-Overview_of_a_three-tier_application_vectorVersion.svg.png

en.wikipedia.org/wiki/Standalone_software

Quick Reference Sheet Links

http://www1.salary.com/Systems-Architect-salary.html
http://www.payscale.com/research/US/Job=Systems_Architect/Salary
http://www.indeed.com/salary/Solution-Architect.html
https://www.glassdoor.com/Salaries/solutions-architect-salary-SRCH_KO0,19.htm
http://www.payscale.com/research/US/Job=Solutions_Architect/Salary
http://www1.salary.com/Application-Systems-Architect-Salaries.html
http://www.payscale.com/research/US/Job=Technical_Architect/Salary
https://www.glassdoor.com/Salaries/technical-architect-salary-SRCH_KO0,19.htm
http://www.indeed.com/salary/q-Technical-Architect-l-United-States.html