Article

Are You a Software Architect?

Posted by Simon Brown on Feb 09, 2010
Community Architecture Topics Delivering Quality , Leadership , Enterprise Architecture Tags Testing , Architecture Evaluation , Coaching and Mentoring , Requirements

***Editors note**: The author of this article, Simon Brown, will be presenting a tutorial at QCon London coming up this March on the same subject of this article, entitled Software Architecture for Developers.*

The line between software development and software architecture is a tricky one. Some people will tell you that it doesn't exist and that architecture is simply an extension of the design process undertaken by developers. Others will make out it's a massive gaping chasm that can only be crossed by lofty developers who believe you must always abstract your abstractions and not get bogged down by those pesky implementation details. As always, there's a pragmatic balance somewhere in the middle, but it does raise the interesting question of how you move from one to the other.

Some of the key factors that are often used to differentiate software architecture from software design and development include an increase in scale, an increase in the level of abstraction and an increase in the significance of making the right design decisions. Software architecture is all about having a holistic view and seeing the bigger picture to understand how the software system works as a whole. While this may help to differentiate software development and software architecture, it doesn't necessarily help to understand how somebody moves from development into architecture. Furthermore, it also doesn't help in identifying who will make a good software architect, how you go about finding them if you're hiring and whether *you* are a software architect.

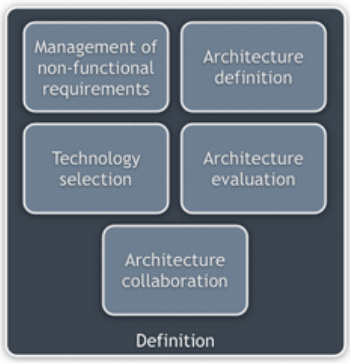
Experience is a good gauge but you need to look deeper

Becoming a software architect isn't something that simply happens overnight or with a promotion. It's a role, not a rank. It's an evolutionary process where you'll gradually gain the experience and confidence that you need to undertake the role.

There are a number of different qualities that you can look for in a software architect and their past experience is often a good gauge of their ability to undertake the role. Since the role of a software architect is varied though, you need to look deeper to understand the level of involvement, influence, leadership and responsibility that has been demonstrated across a number of different areas. Broadly speaking, the software architecture on most projects can be broken down into two phases; the architecture is defined and then it's delivered.

Definition of the software architecture

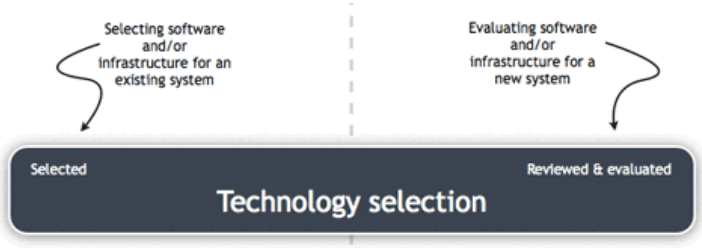
The architecture definition process seems fairly straightforward. All you have to do is figure out what the requirements are and design a system that satisfies them. But in reality it's not that simple and the software architecture role can vary wildly depending on how engaged you are and how seriously you view your role. As the following diagram shows, the architecture definition part of the role can be broken down further into a number of different elements.



1. **Management of non-functional requirements**: Software projects often get caught up on asking users what features they want, but rarely ask them what non-functional requirements (or system qualities) they *need*. Sometimes the stakeholders will tell us that "the system must be fast", but that's far too subjective. Non-functional requirements need to be specific, measurable, achievable and testable if we are going to satisfy them. Most of the non-functional requirements are technical in nature and often have a huge influence on the software architecture. Understanding the non-functional requirements is a crucial part of the role, but there's a difference between assuming what those requirements are and challenging them. After all, how many systems have you seen that genuinely need to be operational 24x7?

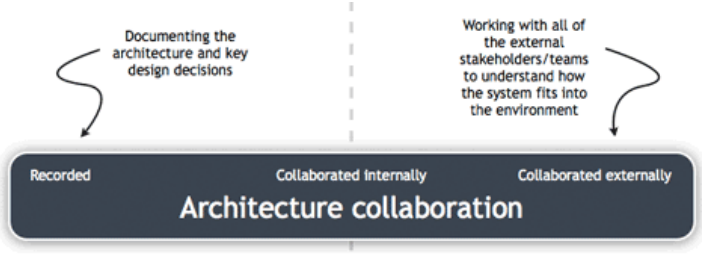
2. **Architecture definition**: With the non-functional requirements captured, the next step is to start thinking about how you're going to solve the problems set out by the stakeholders and define the architecture. It's fair to say that every software system *has* an architecture, but not every software system has a *defined* architecture. And that's really the point here. The architecture definition process lets you think about how you're going to take the requirements along with any imposed constraints and solve the problem. Architecture definition is about introducing structure, guidelines, principles and leadership to the technical aspects of a software project. Defining architecture is your job as a software architect but there's a big difference between designing a software system from scratch and extending an existing one.

3. **Technology selection**: Technology selection is typically a fun exercise but it does have its fair set of challenges when you look at cost, licensing, vendor relationships, technology strategy, compatibility, interoperability, support, deployment, upgrade policies, end-user environments and so on. The sum of these factors can often make a simple task of choosing something like a rich client technology into a complete nightmare. And then there's the question of whether the technologies actually work. Technology selection is all about managing risk; reducing risk where there is high complexity or uncertainty and introducing risk where there are benefits to be had. Technology decisions need to be made by taking all factors into account, and all technology decisions need to be reviewed and evaluated. This includes the major building blocks for a software project right down to the libraries and frameworks being introduced during the development. If you're defining an architecture, you also need to be confident that the technology choices being made are the right choices. Again there's a big difference between evaluating technology for a new system versus adding technology into an existing system.



4. **Architecture evaluation**: If you're designing software, you need to ask yourself whether your architecture will work. For me, an architecture works if it satisfies the non-functional requirements, provides the necessary foundation for the rest of the code and provides a sufficient platform for solving the underlying business problem. One of the biggest problems with software is that it's complex and abstract, which makes it hard to visualise the runtime characteristics from UML diagrams or the code itself. We undertake a number of different types of testing throughout the software development lifecycle to give us confidence that the system we are delivering will work when rolled out. So why don't we do the same for our architectures? If you can test your architecture, then you can prove that it works. And if you can do this as early as possible, you can reduce the overall risk of project failure rather than simply hoping for the best.

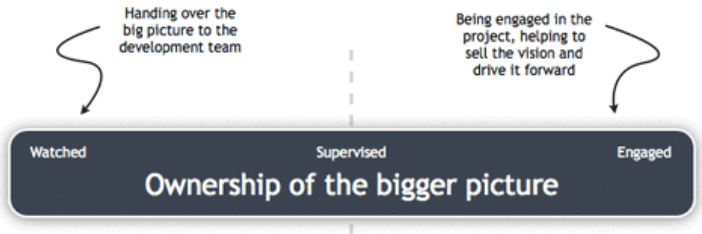
5. **Architecture collaboration**: It's unusual for any software system to live in isolation and there are a number of people that need to understand it. This ranges from the immediate development team who need to understand and buy in to the architecture, right through to other stakeholders who have an interest from a security, database, operations, maintenance, support, etc point of view. For a software project to be successful, you need to collaborate closely with all of the system stakeholders to ensure that the architecture will successfully integrate with its environment. Unfortunately, architecture collaboration within the development team seldom happens, let alone the external stakeholders.



Delivery of the software architecture

It's the same story with architecture delivery too, where the software architecture role can vary depending on the level of engagement across the elements that contribute to a successful software project.

1. **Ownership of the bigger picture**: In order to carry the architecture through to a successful conclusion, it's important that somebody owns the big picture and sells the vision throughout the entirety of the software development lifecycle, evolving it throughout the project if necessary and taking responsibility for ensuring that it's delivered successfully. If you've defined an architecture, it makes sense to remain continually engaged and evolve your architecture rather than choosing to hand it off to an "implementation team".



2. **Leadership**: Owning the bigger picture is one aspect of technical leadership, but there are other things that need to be done during the delivery phase of a software project. These include taking responsibility, providing technical guidance, making technical decisions and having the authority to make those decisions. As the architect, you need to undertake the technical leadership to ensure everything is taken care of and that the team is being steered in the right direction on a continuous basis. The software architect position is inherently about leadership and while this sounds obvious, many project teams don't get the technical leadership that they need, with architects assuming that a successful delivery isn't necessarily their problem.

3. **Coaching and mentoring**: Coaching and mentoring is an overlooked activity on most software development projects, with many team members not getting the support that they need. While technical leadership is about steering the project as a whole, there are times when individuals need assistance. In addition to this, coaching and mentoring provides a way to enhance people's skills and to help them improve their own careers. This is something that should fall squarely within the remit of the software architect, and clearly there's a big difference between coaching your team in architecture and design versus helping them with their coding problems.

4. **Quality assurance**: Even with the best architecture and leadership in the world, poor delivery can cause an otherwise successful project to fail. Quality assurance is a large part of an architect's role, but it's more than just doing code reviews. For example, you need a baseline to assure against, and this means the introduction of standards and working practices. From a software development perspective, these could include coding standards, design principles and source code analysis tools through to the use of continuous integration, automated unit testing and code coverage tools. It's safe to say that most projects don't do enough quality assurance, and therefore you need to figure out what's important and make sure that it's sufficiently assured. For me, the important parts of a project are anything that is architecturally significant, business critical, complex or highly visible. You just need to be pragmatic and realise that you can't necessarily assure everything, doing something rather than doing nothing.

5. **Design, development and testing**: The last thing that falls squarely within the role of a software architect is design, development and testing. Being a hands-on architect doesn't necessarily mean that you have to get involved in the day-to-day coding tasks, but it does mean that you're continuously engaged in the project, actively helping to shape and deliver it. Having said that, why shouldn't the day-to-day coding activities be a part of an architect's role? Most architects are experienced coders, so it makes sense to keep those skills up-to-date. In addition, the architect can experience the same pain as everybody else on the team, which in turn helps them better understand how their architecture is viewed from a development perspective. Many companies have policies that prevent software architects from engaging in coding activities because their architects are "too valuable to undertake that commodity work". Clearly this is the wrong attitude ... why let your architects put all that effort into defining the architecture if you're not going to let them contribute to its successful delivery? Of course, there are situations where it's not practical to get involved at the code level. For example, a large project generally means a bigger "big picture" to take care of and there may be times when you just don't have the time. But generally speaking, an architect that codes is more effective and happier than an architect that watches from the sidelines.

Are you a software architect?

Regardless of whether you view the line between software development and architecture as mythical or a gaping chasm, the elements above highlight that people's level of experience across the software architecture role varies considerably depending on how engaged they are and how seriously they view their role. Most developers don't wake up on a Monday morning and declare themselves to be a software architect. I certainly didn't and my route into software architecture was very much an evolutionary process. Having said that though, there's a high probability that those same developers are *already* undertaking parts of the software architecture role, irrespective of their job title.

There's a big difference between contributing to the architecture of a software system and being responsible for defining it yourself; with a continuum of skills, knowledge and experience needed across the different areas that make up the software architecture role. Crossing the line between software developer and software architect is up to you, but understanding your own level of experience is the first part of the journey.

About the Author

Depending on your viewpoint, Simon Brown is either a software architect who codes or a software developer who understands architecture. When he's not developing software with .NET or Java, Simon can usually be found consulting, coaching or training. Simon has also written books about Java, presented at industry events and has put together a training course called Software Architecture for Developers, which is based upon his software architecture writing at Coding the Architecture. You can catch up with Simon by e-mail or Twitter.

28 comments Watch Thread Reply

Good Article by Colin Song Posted Feb 9, 2010 5:57 AM
You just need to be pragmatic and realise that you can't necessarily. by Paulo R. A. Sales Posted Feb 9, 2010 8:36 AM
Good Article by Alexandre Poitras Posted Feb 9, 2010 10:21 AM
Re: Good Article by Simon Brown Posted Feb 10, 2010 3:49 PM

Sort by date descending

Good Article

Feb 9, 2010 5:57 AM by **Colin Song**

#, Thanks

Reply

You just need to be pragmatic and realise that you can't necessarily.

Feb 9, 2010 8:36 AM by **Paulo R. A. Sales**

Hi,

First all congratulations!

I like your view about a software architect which can code with team to keep their skills and feel how your architecture is hard or easy to implement. I think that is realy necessary a software architect act this form.

Thank

Reply

Good Article

Feb 9, 2010 10:21 AM by **Alexandre Poitras**

Couldn't Have Said It Better! The problem most of the time is having to deal with architects who are not “hand's on” and are more of a drag the team. Getting ride of a software architect or a technical lead or call them whatever you want is a sure way to a complete mess in any kind of "big" software projects. Been there, done that. I think the solution is to give the team members a way to "reject" a bad architect.

Reply

Love the emphasis on Ownership, Collaboration, Leadership

Feb 9, 2010 11:25 AM by **Tim Morrow**

Excellent article. We at Shopzilla would agree that Ownership, Collaboration and Leadership are qualities as important as the more well-understood areas of non-functional requirements gathering, architecture definition and technology selection. We attempted to define a set of aspirational qualities that describe the Software Architect role entitled I am an Architect

Reply

Great. Remember Architecture is the final product

Feb 9, 2010 11:59 AM by **William Martinez**

Nice summary, Simon.
I would remark that Architecture is not the design, but the working structure after the coding is complete. That is, the architect should not end his job without seen the creature crawling.
The architecture description and definition is all that you say.
Architect is a part of the team, not superior, and may not be expert on the language at hand, but should be involved in coding. And coding does

not mean writing ifs and classes, but actually deciding on strategic, tactical and operational issues.

Cheers!

William Martinez Pomares.
Architect's Thoughts Blogger Twitter

Reply

Quite in Line

Feb 9, 2010 12:19 PM by **Vikas Hazrati**

Great points and most of them are quite in line with the characteristics of an Agile Architect that I wrote some time back.

Reply

Good Article

Feb 9, 2010 10:40 PM by **Maqdoom Mohammed**

Fantastic collection of ideas and presentation. Thanks a lot.

Reply

Is it really that simple?

Feb 9, 2010 11:42 PM by **Gary Shepard**

Very nice article, However I believe that architecture is not as simple as some of the comments make it out to be, what I see described here is Application Architect, but what happens when you dealing with a large sale enterprise, with multiple applications and multiple projects, with multiple development teams, and multiple technologies, there are integration points to deal with, technology baselines, standards that are more than coding idioms, at this point architecture take on a whole new meaning. Consider the National Institute of Health Enterprise Architecture Model, certainly an architect with “hand on” experience could not possibly have the bandwidth to participate in coding effort.

When it comes to the title of "Architect" there must be a distinction as what type of architect do you need!

Reply

Integration Architecture?

Feb 10, 2010 12:34 AM by **Hamdi Yusof**

Nice article.

One question please.

What about integration architecture which integrates dozens of systems (java, .net, .net mobile running in scanners, vb6, sap, etc) fairly seamlessly. Are these the work of a software architect?

Thanks.

Reply

What about functional requirements?

Feb 10, 2010 5:29 AM by **Johannes Brodwall**

Nice overview of some of the concerns of a software architect. But when did we manage to define as out of the architect's scope *what we're going to build*, that is, the functional requirement? If there's to be any meaning to using the metaphor of an architect, we should be concerned with what function our "building" is supposed to serve, no?

Reply

Interesting article

Feb 10, 2010 6:01 AM by **Thanh-Liem Tran**

First of all, I would like to congratulate you for your article.

In my point of view, a good architecture design must be extensible and must also be able to fit into an existing system. The architecture solution proposed must response to the needs of the client (which are most of the time funky), and respect the non-functional aspect of the application without impacting the scalability and the quality of the whole system (non-regression).

Choosing the technology is also, an important aspect of the role played by the architect. In most of the time, people will goes with the technology they knows best, others will go with "extreme" technology, which are most of the time hard to find resources to maintain this techno. Having a compromise, to choose a technology that can be use is another debate. But the number of different technologies used within a company are most of the time limit, and it is normal.

In conclusion, the architect role, is not as simple as it appear.

cheers.

Reply

Good article. Clear and concise

Feb 10, 2010 6:19 AM by **Juan Jose Zapico**

I agree with all the items you point out in "Delivery of the software architecture". I think that in "Ownership of the big picture" also should be emphasized the communication skills needed to involve everyone that could be affected in some way by the project.
Cheers.

Reply

It's all about the Abstractions, baby

Feb 10, 2010 8:27 AM by **Dave Marney**

Don't disagree with any of the team building / leadership / ownership / eat-your-own-dogfood aspects highlighted by the article, but aren't those the qualities of good senior people in general, not necessarily those of architects in particular?

Here's my pick for what sets architects apart:

Architects have the ability to think abstractly, to recognize patterns outside of their immediate context, and apply them in novel situations.

They operate primarily from first principles instead of instances.

They are experimental, creative, and iterative in how they approach solving problems; the first solution is rarely the best answer, so they keep digging until they hit the gold.

They are gifted researchers, constant learners, and incisive thinkers. They truly know their craft.

They are excellent communicators. They listen as well as they talk, they get to the point of a matter, they have useful and insightful ways of explaining things, and they keep at it until everyone truly "gets it".

A pretty hard set of skills to find in one person.

Reply

Re: Is it really that simple?

Feb 10, 2010 8:33 AM by **Hamza BENMANSOUR**

Very good article and equally good comment. I think that when it comes to integration, we can introduce a new role, information system architect, who is in charge of defining the way different applications in the system interact, including the synchronization of data, the definition of exchange format, the defintion of domains borders, ...

Reply

Great Article

Feb 10, 2010 8:58 AM by **Fernando Franzini**

Great article. Congratulations.

Reply

It's really about a bigger picture

Feb 10, 2010 9:13 AM by **Shirley Z**

Architect at various levels many not have to write code, but an architect must have a vision that is bigger than their team individual. And as someone else pointed out here - ABSTRACTION - the capability of working out an architectural solution without having to get every pieces of details from the customer.

Reply

Re: Integration Architecture?

Feb 10, 2010 9:18 AM by **Paul Nita**

Absolutely! You may call this role Integration Architect or just System Architect, it still requires a lot of architecture skills to integrate disparate systems and technologies.

Reply

SCEA

Feb 10, 2010 12:07 PM by **roberto parra**

a Monday morning I wake up, i take and pass the SCEA exam and they declare me ARQUITECT!!! :S

Reply

scale

Feb 10, 2010 1:47 PM by **Humphrey Bogart**

Head up Architecture for an organization with a $12bn IT budget, 7,000 business apps, 0.25m users, 1bn clients, and then try to lead by coding.

Reply

Re: scale

Feb 10, 2010 3:10 PM by **Simon Brown**

That's a very, very different role than "software" architecture! :-)

Reply

Re: It's all about the Abstractions, baby

Feb 10, 2010 3:42 PM by **Simon Brown**

> aren't those the qualities of good senior people in general, not necessarily those of architects in particular?

I agree, indeed they are and I think your additional list of skills can also be applied to any good senior people, non-technical roles included. I alluded to this at the end of the article, but there are lots of senior people out there that are already doing what we've both described and aren't recognised for it. On the other hand, there are lots of people out there with "architect" in their job title that don't necessarily have those qualities.

> A pretty hard set of skills to find in one person.

It is, and I've seen people that haven't been recognised in the past because they've not "ticked all of the boxes". My view of the software architect role came about from interviewing people and trying to ascertain what level of experience and engagement they *really* had on their projects. It turned a "box ticking" exercise into a discussion, which I found really useful in helping to identify whether they were a software architect.

Reply

Re: Good Article

Feb 10, 2010 3:49 PM by **Simon Brown**

Thanks very much.

You're right, and it would be great if organisations used something like the model I described when reviewing their software architects rather than the cursory, "yes, you seem to have done a good job". Perhaps teams should implement the same agile feedback mechanism that the QCon organisers do ... at the end of every iteration/release everybody votes with a green, yellow or red card for each team member. Get too many red cards and you're out! :-)

Reply

Re: Great. Remember Architecture is the final product

Feb 10, 2010 4:15 PM by **Simon Brown**

Thanks William.

> I would remark that Architecture is not the design, but the working structure after the coding is complete. That is, the architect should not end his job without seen the creature crawling.

I agree and having a "working" structure is why it's crucial for software architects to be involved in defining it and delivering (building and using) it. "Working", for me, means delivering the right foundations to provide business value and seeing the creature crawl (preferably as early as possible).

> Architect is a part of the team, not superior

That's an interesting point ... software architecture is really just a different role played out on a software team. To be honest, it doesn't need to be allocated to a single person and I'd be really happy to see a team where everybody is collaborating on the architectural aspects of the software. Unfortunately it's more common to see teams where nobody is thinking about it, and that's when you get systems that don't perform/scale/fit into the operational environment/etc. We'll build better software as an industry if everybody better understands the "bigger picture" stuff.

Reply

Re: What about functional requirements?

Feb 10, 2010 4:21 PM by **Simon Brown**

Good point. :-) Software architecture most definitely needs to be defined and delivered in the context of the functional requirements. I've certainly been caught out by not keeping the functional requirements in mind and Udi Dahan has a great post on this very issue ... www.udidahan.com/2010/01/12/non-functional-arch...

Reply

Re: It's all about the Abstractions, baby

Feb 10, 2010 4:26 PM by **Rainer Eschen**

There's a lot of truth in your list. Communication is still the most important one to me:

blog.rainer.eschen.name/2008/02/20/the-mission-...

Reply

Good One, reflection of my view

Feb 11, 2010 6:00 AM by **Binish Peter**

good one Simon,
exact reflection of my view.

/binish

Reply

Re: It's really about a bigger picture

Feb 12, 2010 1:23 AM by **Salih VARLI**

Architect at various levels many not have to write code, but an architect must have a vision that is bigger than their team individual. And as someone else pointed out here - ABSTRACTION - the capability of working out an architectural solution without having to get every pieces of details from the customer.

Sesli Sohbet
Turk Sohbet

Reply

Re: Integration Architecture?

Feb 12, 2010 1:23 AM by **Salih VARLI**

Absolutely! You may call this role Integration Architect or just System Architect, it still requires a lot of architecture skills to integrate disparate systems and technologies.

Sesli Sohbet
Turk Sohbet

Reply