The Importance of Software Architecture

Since architecture is a vital part of any software development process, business leaders should understand its purpose and value before hiring a development firm. Proper architecture is a system framework that, among other things, guides development efforts and helps to reduce the overall cost of software. In Agile development, some of the key architectural decisions are made during an initial planning stage but the complete architecture is actually developed incrementally, sprint by sprint. Drawing on business and technical expertise, creative ability, and well-honed people skills, a software architect collaborates with all stakeholders to transform requirements into great software that works. This white paper describes software architecture and explains why it's so important.

www.architech.ca

Introduction

...architecture decisions are important if your system is going to meet its quality requirements and...you want to pay attention to the architecture and make these decisions intentionally rather than just "letting the architecture emerge."

- Beautiful Architecture[1]

Architecture, like life, is all about the people. Great people deliver great architecture, and the reverse is true.

- James McGovern, etc
A Practical Guide to Enterprise Architecture[2]

The construction of a software system usually does not end in delivery of the product—a software system also evolves with time. During evolution, often new features are added to the system. The architecture of the system can help in deciding where to add the new features with minimum complexity and effort, and what the impact on the rest of the system might be of adding the new features.

- Pankaj Jalote
An Integrated Approach to Software Engineering[3]

Software is complex. The average residential home is not. Why, then, is it acceptable to start building software without an architectural plan but entirely unacceptable to approach home construction in a similarly haphazard fashion? Answer: It's not.

Consider these six statements about software development:

1. Software is derived from a set of clearly defined business requirements.
2. Software is intended to meet the needs of the business and work the way end-users work.
3. Software must be able to accommodate business requirements that inevitably change over time. For example, the number of users might be small initially but later grow to a large number, which means that the software must scale accordingly.
4. Business processes and corresponding software systems are complex.
5. Software components need to fit together seamlessly.
6. Best practices, properly employed, can help make software easier to upgrade, expand, scale, etc. Also, use of best practices reduces the likelihood that a business will be locked in to a particular vendor.

Based on these statements, it's easy to conclude that software can't just be thrown together and expected to work. It's too complicated. Rather, it must be carefully designed, or engineered, for a specific purpose.

Another reasonable conclusion is that most business leaders can't be expected to define a complete set of requirements, either because they lack the relevant experience and expertise, or because they simply don't know all the requirements up front.

Instead, it makes sense for them to collaborate with a knowledgeable technology partner that has a keen understanding of business and computer systems, and a proven ability to translate business requirements into great software.

"Engineering" and "architecture" aren't just empty buzz words, or marketing speak for "expensive, high-brow programming". Proper architecture is smart planning, and it actually brings down the cost of software and reduces product development risk.

To make informed decisions about their software projects, business leaders must clearly understand the role of architecture in the development process. This white paper describes software architecture and explains why it is critical to building great software that works.

Architecture Explained

Software architecture has been described in many different ways—indeed, every book on the subject seems to include a highly technical definition of some sort—but the basic idea can be summarized quite simply. **An architecture is a plan that describes significant systems decisions from a variety of perspectives, all of which are aligned with business goals.**

The plan typically outlines the overall system structure, describes its various components, indicates how the components fit together, and defines the rules and standards that govern their behaviour. The goal of software architecture is to focus development resources on building the right software for a given situation, with as little wasted effort as possible.

Fortunately, the architect doesn't have to pull a plan out of thin air. The choice of architecture is based on the architect's experience, the development team's experience, best practices, and the inherent constraints of the project. Done well, architecture is a highly collaborative effort that draws upon the knowledge and skill of an entire team.

In software development, changes are inevitable and flexibility is a key element of any architecture. As the architect and the development team progress through development iterations (sprints) and learn more about the project, they repeatedly fine-tune the requirements and technology. Architecture should point the way forward, offer a working framework, but remain fluid enough to be adapted as circumstances dictate.

One way to look at architecture is that it, along with analysis, is the bridge between requirements and design. Architects will also consider high-level design decisions such as "What will we build this system from?", "How will the parts of this system fit together?", and "How can we ensure this system will work under less-than-perfect (or real-world) conditions?"

- Scott W. Ambler
The Object Primer[4]

Guiding Principles

Proper architecture is the best solution that simultaneously satisfies business needs, business constraints (including cost), and technical constraints. To achieve the right balance between these competing interests, the architect must have a good understanding of four key areas:

1. **The business**
   The company's mission statement, philosophy, core values, competencies, business strategy, and activities.
2. **Processes and existing computer systems**
   The way that end-users accomplish their work and the hardware/software that they are currently using.
3. **Project constraints**
   What the business wants the software to accomplish, hardware/software that the business is locked into using, timeline, and budget.
4. **Technology**
   Suitability, cost, and compatibility with existing systems.

Software architectural styles are established, large-scale patterns of system structure. Analogous to architectural styles for buildings, software architectural styles have defining rules, elements, and techniques that result in designs with recognizable structures and well-understood properties. However, styles are not complete detailed solutions. Rather, they are loose templates that offer distinct solutions for coordinating a system's components. To be specific, architectural styles focus on the different ways that components might communicate, synchronize, or share data with one another.

- Pfleeger & Atlee
Software Engineering:
Theory and Practice[5]

Describing Architecture

Software architecture can be very complicated and therefore difficult to describe as a whole. To solve this problem, architects divide up various aspects of the architecture into categories, or views, and then construct an

overall architectural model using these views. Each view describes a particular subset of architectural concerns.

For example, a “physical” view might describe the hardware aspects of the architecture, like servers, storage devices, and the network. One model for describing software architecture is the 4+1 View Model.

An architecture description is primarily to communicate the architecture to its various stakeholders, which include the users who will use the system, the clients who commissioned the system, the builders who will build the system, and of course, the architects.

- Pankaj Jalote
An Integrated Approach to Software Engineering[6]

By representing the architecture in this modular way, architects make it easy for all stakeholders in a development project to understand the aspects that concern them without having to sort through a lot of extraneous information.

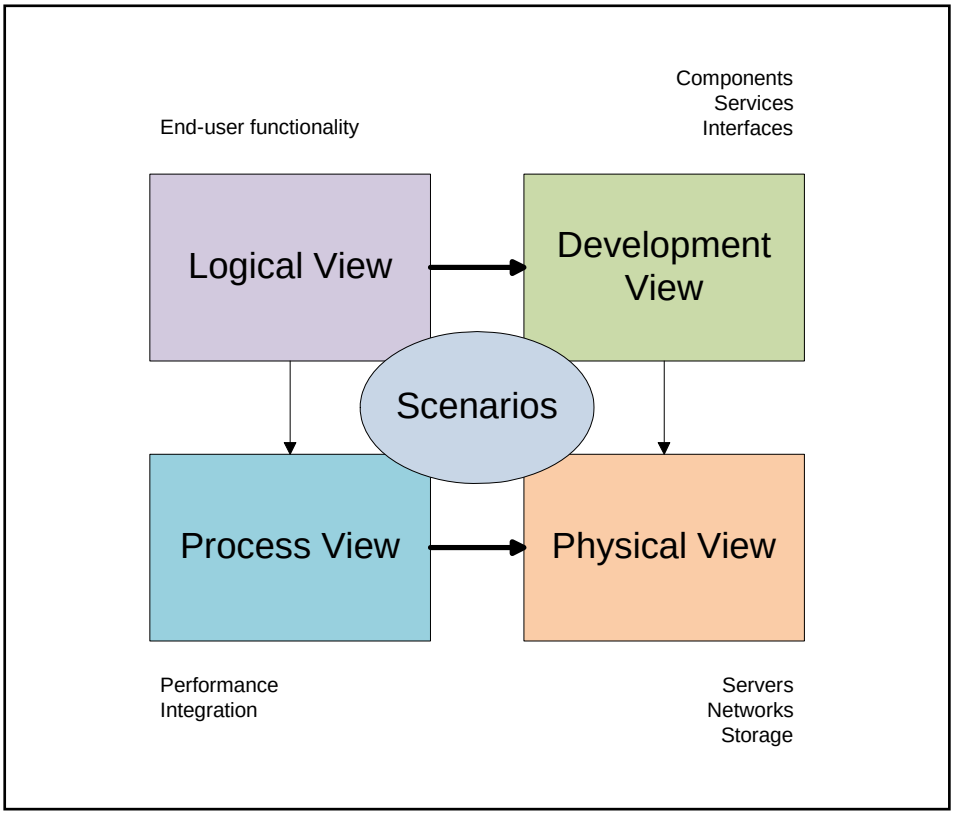

Figure: 4+1 View Model of Software Architecture

In cartography, a single map cannot fully characterize a place. There are many kinds of maps, such as highway maps, bike trail maps, and elevation maps. Each type explains and describes different aspects of the same physical place. Each map is relevant to a different user or stakeholder...The same holds true in software architecture.

- James McGovern, etc
A Practical Guide to Enterprise Architecture[7]

Indicators of Proper Architecture

Proper architecture helps to produce software that works. Therefore, architectural quality can be gauged by measuring the quality of the finished software using common attributes like:

- **User friendliness**
  How easy is it to use the software?
- **Functionality**
  Does the software work the way end-users work?
- **Robustness**
  How reliably does the software run in the real world?

One of the more interesting problems in designing an architecture for a system is ensuring flexibility in the scale of the system. Scaling is becoming increasingly important, as more of our systems are run on networks or are available on the Web.

- Beautiful Architecture[8]

- **Scalability**
  How well does the software handle increasing numbers of transactions, users, etc?
- **Flexibility**
  To what extent can the software be adapted by the users themselves (without the need for IT intervention)?
- **Extensibility**
  How easily can software functionality be expanded or modified?
- **Performance**
  How quickly does the software run?
- **Security**
  Are there any flaws that would permit unauthorized access to the system?
- **Testability**
  Is it easy to run meaningful tests on the software?
- **Maintainability**
  How easily can the software be maintained?

Software with a weak architecture is inevitably deficient in one or more of these key attributes.

Management, in general, views an information system as a tool for achieving business objectives quickly and effectively, for both the short term and the long term. Managers need to see that flexible systems allow faster and less expensive reaction to change.

- Bruce Johhnson, etc
Flexible Software Design[9]

The Importance of Keeping Options Open

In Agile development, decisions—even architectural ones—are delayed as long as possible to maximize the amount of information available and to accommodate changing business needs. Gradually, the team locks in more and more of the architecture but the longer they can keep their options open, the easier it will be to respond to unexpected changes.

During the initial planning stage of a software project, the architect should commit to just enough architecture to get the development team started on its first sprint. For example, programmers will need to know what language to use before they can begin writing code. In each sprint, additional architectural specifications (aka non-functional requirements) are communicated to the team in the form of *architecture user stories* (aka non-functional user stories). For example, a user story might read, “As a client, I need my software to run on Windows XP and Windows 7 operating systems.” The user story describes a software constraint, not a software feature.

Characteristics of a Good Software Architect

Architecture is as much an artistic and creative endeavour as an engineering one...

- Pfleeger & Atlee
Software Engineering: Theory and Practice[10]

A software architect has to be one part technical expert, one part politician, and one part visionary/evangelist. An experienced programmer and software engineer, the architect collaborates with businesspeople and developers alike to translate business requirements into the best possible technical framework.

The architect develops a software architecture by working through a series of calculated trade-offs between attributes such as scalability, performance, and

maintainability. The architect strives to find the optimal *balance* of these attributes within the constraints of budget, resources, and market considerations. A skilful architect has a knack for achieving and maintaining this balance, despite the multitude of variables in play.

To properly configure such a complex set of factors in a way that satisfies business objectives, the architect must be able to see the system from multiple levels of abstraction (models) and from numerous perspectives (views). A conduit between business and IT, and a leader, coach, and mentor to the development team, the architect must have the people skills to keep a project moving toward successful completion.

As a creative thinker with a big-picture consciousness and a passion for great software, the architect must formulate an overall vision for the project that will guide development. Equally important, the architect should make well-reasoned decisions and then clearly rationalize those decisions to a broad range of stakeholders. To do this effectively, the architect must be a down-to-earth communicator who never broadcasts from an ivory tower.

No architecture is perfect. Requirements often contradict each other, priorities change, and sometimes you may find that you simply do not have the resources to do everything that you would like. The point is that you cannot always get what you want and that you need to be prepared to make trade-offs as you architect a system.

- Scott W. Ambler
The Object Primer[11]

Challenges to Proper Architecture

In pursuing this hybrid discipline that melds business, technical, and people skills, the software architect is often called upon to perform a delicate high-wire act. Here are some of the challenges that the architect must overcome to protect the integrity of the architecture and produce great software:

- **Politics**
  The software architect is a key interface between business and IT, and therefore must weather the storms that brew along that front. The architect must act as a mediator, helping to reach a consensus on what needs to be accomplished and how it should be done. Some compromises, while necessary, are problematic from an architectural standpoint.
- **Out-of-date standards**
  Existing software components might conform to old standards, making it difficult for an architect to integrate them into a new system.
- **Exclusive focus on technology**
  If an architect compromises business needs to achieve a specific technical outcome, the resulting architecture will be inappropriate and ineffective.
- **Poor understanding of the business goals or domain**
  The architecture cannot succeed unless the architect really understands the business that must be served by the software.
- **Lack of understanding by the business**
  A business that does not understand the value of proper software architecture will push for shortcuts at the planning stage, which will lead to “accidental” architecture and the “shanty town” software that goes with it.
- **Poor implementation**
  The most brilliant design is useless without the ability to act on it. The architect must have the skills, support, and resources to shepherd a software project from requirements to successful release.

Architecture models act as bridges between the business and the technical sides of your project. A model that focuses primarily on business issues, or on technical issues, will not meet your real-world needs. When business stakeholders or IT professionals do not see concepts that they can understand, and mapping of those concepts to the ideas that they may not immediately comprehend, they will soon abandon the architecture efforts. You need to find the model that meets everyone's needs.

- James McGovern, etc
A Practical Guide to Enterprise Architecture[12]

References

1. Spinellis, Diomidis, and Georgios Gousios (editors). *Beautiful Architecture: Leading Thinkers Reveal the Hidden Beauty in Software Design,* (Sebastopol, California: O'Reilly Media, 2009), 12.
2. McGovern, James, Scott W. Ambler, Michael E. Stevens, James Linn, Vikas Sharan, and Elias K. Jo. *A Practical Guide to Enterprise Architecture,* (Upper Saddle River, New Jersey: Pearson Education, 2004), 37.
3. Jalote, Pankaj. *An Integrated Approach to Software Engineering (Third Edition)*, (New York: Springer Science+Business Media, 2005), 162-163.
4. Ambler, Scott W. *The Object Primer: Agile Model-driven Development with UML 2.0 (Third Edition),* (New York: Cambridge University Press, 2004), 278.
5. Pfleeger, Shari Lawrence, and Joanne M. Atlee. *Software Engineering: Theory and Practice (Fourth Edition),* (Upper Saddle River, New Jersey: Pearson Higher Education, 2010), 236.
6. Jalote, *An Integrated Approach to Software Engineering,* 161.
7. McGovern, *A Practical Guide to Enterprise Architecture,* 37.
8. Spinellis, *Beautiful Architecture,* 45.
9. Johnson, Bruce, Walter W. Woolfolk, Robert Miller, and Cindy Johnson. *Flexible Software Design: Systems Development For Changing Requirements,* (Boca Raton, Florida: Auerbach Publications, 2005), 51.
10. Pfleeger, *Software Engineering,* 289-90.
11. Ambler, *The Object Primer,* 285.
12. McGovern, *A Practical Guide to Enterprise Architecture,* 152.

Architech Solutions
3 Church Street
Suite 602
Toronto, Ontario
M5E 1M2

Phone: (416) 607-5618
Fax: (416) 352-1768