

# Is Software Architecture a REAL Profession?

A commonly-held view of software architects is that they operate apart from programmers, do not understand the issues that programmers deal with, and have little impact on the realities of software projects.

Another view is that software architects play a critical role in ensuring that a project is a success, and is perhaps the most critical member of a software development team.

Which view is correct? Is the answer universal? Let's first clarify what a software architect really is.

## Bureau of Labor Statistics Definition

According to the U. S. Bureau of Labor Statistics (BLS), there is no such job as "software architect". The most closely related job categories include Computer Software Engineer, Computer Systems Analyst, Computer Scientist, Database Administrator, Computer Programmer, Computer Support Specialist, and Systems Administrator.

From the BLS website, Computer Software Engineers "... are more concerned with developing algorithms and analyzing and solving programming problems than with actually writing code."

According to BLS, Computer Programmers "... write, test, and maintain the detailed instructions, called programs, that computers must follow to perform their functions." The job description goes on to say, " Many technical innovations in programming—advanced computing technologies and sophisticated new languages and programming tools—have redefined the role of a programmer and elevated much of the programming work done today. Job titles and descriptions may vary, depending on the organization."

What about Computer Scientist? BLS states that Computer Scientists "work as theorists, researchers, or inventors." Computer Systems Analysts "solve computer problems and apply computer technology to meet the individual needs of an organization...they also may prepare cost-benefit and return-on-investment analyses to help management decide whether implementing the proposed technology will be financially feasible."

Is it possible that one of these categories includes what we commonly refer to as a "software architect"? Not likely, because a research report [Pikulinski2004] that is published on the BLS website mentions the job category "Chief Software Architect" as a new type of job in the Services sector. But to really answer this question, we have to come up with our own understanding of what a software architect is – or should be. For this purpose, let's consider an analogous profession that has a clearly defined profession of architect: building.

Going back to the BLS, "[Building] Architects design the overall aesthetic and

look of buildings and other structures, but the design of a building involves far more than its appearance. Buildings also must be functional, safe, and economical and must suit the needs of the people who use them. Architects consider all these factors when they design buildings and other structures....Architects...may be involved in all phases of development, from the initial discussion with the client through the entire construction process. Their duties require specific skills—designing, engineering, managing, supervising, and communicating with clients and builders. Architects spend a great deal of time explaining their ideas to clients, construction contractors, and others. Successful architects must be able to communicate their unique vision persuasively."

In the building profession, the role of architect seems to focus on the ability of an individual to look at the big picture, and ensure the technical success of a project. It involves ensuring that important requirements are captured; for example, the BLS description says, "Buildings also must be functional, safe, and economical and must suit the needs of the people who use them." It spans all phases of development: "They may be involved in all phases of development, from the initial discussion with the client through the entire construction process." It is specialized: "Their duties require specific skills—designing, engineering, managing, supervising, and communicating with clients and builders," and the last of these emphasizes the ability to communicate with the builders themselves and also with the clients. Thus, it seems that an architect is a very rare and skilled individual: one who has the requisite technical training, who has the ability to see the big picture and anticipate requirements, and who also has the ability to communicate with those implementing the architecture (the builders) as well as the client. This is clearly not only someone with experience: it is someone with training and a natural predisposition to technical leadership.

## **Views of Educators**

Computer science programs tend to focus on algorithms and data structures, rather than on issues that are important for reducing risk in software applications. For example, an informal survey of two local and prominent universities with reputable computer science degree programs revealed that little or no training in important topics such as security and software reliability are required for a degree. This implies that one of the primary values of architecture in the building profession, namely safety and integrity, is not a core value of these computer science training programs. Whether this is true universally is not known, but conversations with educators seem to indicate that it is.

## **Views of IT Managers**

The recognition of the role of software architect across industry varies, even within industry. For example, Fannie Mae does not have such a position, but Freddie Mac does. In most software development shops, programmers are promoted to senior programmer, and then to manager. Large projects often have an architect role that is parallel or superior to lead developer. Some large projects have "architecture teams" of multiple architects who work alongside development teams.

Many software projects have two managers: a business manager and a technical manager who reports to the business manager. The business manager focuses primarily on business issues but must be well-versed in technical issues. The

technical manager focuses on software development process issues and technical issues but must be versed in business issues. The technical manager serves as the primary arbiter of technical decisions, and ensures that appropriate communication and decision processes are organized and performed by the development team. Thus, it seems that the technical manager is performing many of the duties of an architect. In fact, it is possible that the two are synonymous. Perhaps this type of technical manager is actually an architect. If this view is valid, then it provides clarity to the role of technical project manager. It is confusing to have two project managers, one business-focused and one technical. Defining the technical manager as an architect adds precision for what the technical manager is expected to be responsible for.

## **Views of Software Engineering Methodologists: Discussions of What Architecture Is**

Some software methodologists have tackled the question of what software architecture is. For example, Martin Fowler has said, quoting Ralph Johnson from the Extreme Programming mailing list, "Architecture is the decisions that you wish you could get right early in a project." [Fowler2003] In this view, architecture is not a set of diagrams necessarily, or even a specification, but rather is a set of critical decisions that have a large impact downstream. Since decisions imply judgment, and good decisions imply good judgment, this view implies that an architect is someone who can be trusted to make good decisions. Obviously, to be able to make good decisions, one must have acquired good judgment, either through experience, training, or innate ability, and one must also have learned enough about software to be able to recognize patterns that work.

## **Views of Recruiters**

To gauge how widespread the job title "software architect" is within the computing profession, one merely need go online. For example, on Feb. 6, 2006 dice.com listed 6517 positions with a required skill or job title matching the word "architect", but only 212 matching "software architect", out of 33499 containing the word "software". On the same day the ACM website had 114 job postings, none of which were "software architect". Those that mentioned architect at all were Information Architect, Enterprise Architect, and Senior System Architect, with one instance of each of these. Knowing what these titles represent, none of them correspond to a software architect.

Clearly, those who are posting jobs do not generally see "software architect" as a mainstream category. Yet there are other kinds of architect in greater numbers. If these other kinds of architect exist, e. g., data architect and enterprise architect, then why not software architect? Don't good decision-makers matter in software?

## **A Proposed Definition**

We propose a definition for what it means to be a software architect as follows:

A software architect is the primary technical decision-maker for a software project. In this role, the architect ensures that a system's most important requirements are captured, including requirements that impact risk, ensures that the design meets those requirements and also meets cost objectives, ensures

that these requirements are communicated to the builders of the system, arbitrates technical decisions, explains to stakeholders the tradeoffs and how requirements are being met, and has the skills to perform all of these functions.

This borrows directly from the building profession, but *without* borrowing the methodology used by the building profession, which is almost always a waterfall-like process consisting of discrete phases of requirements, design, and implementation. In fact, our definition could easily apply to an individual on an Extreme Programming project as well as it could to a person on a RUP or waterfall project.

## **Software Architecture as a Profession**

It is clear that few programmers meet the criteria for a software architect as we have defined it. It is a rare individual who has the communication skills, the breadth of knowledge, and the soundness of judgment that are required. Yet it is probably the case that any project with significant risk – i. e., in which there is a lot at stake – should have a software architect. The building industry utilizes architects to control risk: a building's architect is the point-person who makes sure that the building is “functional, safe, and economical and must suit the needs of the people who use them.”

If a software architect has rare skills, and those skills are essential for the success of a high-risk project, then it is essential to (a) identify such individuals, (b) ensure they are properly trained and specialized, (c) give them the tools and authority to decide and communicate their decisions, and (d) ensure that they are properly compensated.

Since current educational programs do not address software engineering adequately, it can be concluded that architects are not being properly trained since architecture is a very specialized and higher form of software engineering. Clearly the training in software architecture provided by computer science degree programs is important for an architect, but it is not sufficient. Architects need training in application security, reliability, design, software project economics, communications, and leadership.

The role of communication is interesting. Alistair Cockburn has eloquently pointed out how important communication is for successful software development [Cockburn2000]. Since an architect has to communicate with not only other programmers but also people in other professions, it stands to reason that an architect must be an exceptional communicator in order to be effective. An architect must also have leadership skills in order to retain credibility in a setting in which those who are directly involved in construction will inevitably think they know better by virtue of being closer to the work. In software, this presents a strong argument for an architect to keep his or her “hands dirty” in order to enhance credibility with fellow programmers. That said, there is much debate about whether a software architect should participate directly in code development, or even if an architect should be able to.

Given the current gap between what is common practice and what is needed, it is clear that the industry desperately needs to define the role of software architect, and begin to understand that role in terms of required training and how the role should be utilized in the context of software projects.

One argument against defining software architect is that it might limit what a non-architect is allowed to do. Software is highly driven by grass-roots innovation. A

system that prevents “mere programmers” from having a chance to do “architecture-like” things would be a rigid system that would devolve into bureaucracy. One way to avoid this pitfall is to utilize software development methodologies that encourage all programmers to play an active role in any portion of a system’s design and implementation, but with appropriate supervision. Also, the requirements for one to be an architect should be based on fundamentals, not on current technologies or current methodological approaches; otherwise, over time the architect’s role will become dated. Just as a construction architect must be trained in theoretical fundamentals that are timeless as building technology evolves, a software architect should be trained in the fundamentals that are likely to remain constant as software technology evolves.

We believe that the software engineering profession should define the profession of software architect, and that the definition of that profession should be clear and unambiguous. We also believe that the definition should reflect understanding of software engineering and architecture fundamentals, technical leadership, economic analysis, and communication skills. We believe that software architects are an extremely important segment of the software engineering profession, and should be recognized as such.

## **What Should Be Done?**

In order to establish software architecture as a recognized profession, there must be an accepted understanding of what it means to be a software architect. This paper attempts to provide a basis for discussion on that topic.

As we have seen, the bar for what it takes to be a software architect is very high. This implies that there are many criteria for judging a person’s competence as a software architect. These should include core knowledge about a wide set of issues and subjects, demonstrated sound technical judgment, the ability to communicate with a wide variety of people and to demonstrate leadership, and knowledge of software engineering and estimation, and knowledge of the customer’s business domain. Clearly, to assess all of these things should require a regimen that includes certification in core knowledge and methods for assessing the other skills.

## **Summary**

Different segments of the IT community have disparate views of what a software architect is, and those views are often inconsistent between different organizations of the same kind. There is not even consensus that the role of a software architect differs from that of a technical lead. Yet, there is clearly a need for a highest-level technical decision-maker on complex software projects, and the skills needed by such a person are varied and exceptional.

Software architects need to be recognized for the advanced professionals that they are. This is important not only for software architects, but for an industry that today struggles with the problem of how to build software that is more reliable, secure and scalable, and on which the economy of the future depends.

# References

- Pikulinski2004 "New and Emerging Occupations", by Jerome Pikulinski, Monthly Labor Review, December 2004
- Fowler2003 "Who Needs an Architect?", by Martin Fowler, IEEE Software, July/August, 2003, pp. 2-4
- Cockburn2000 "Characterizing People as Non-Linear, First-Order Components in Software Development", Alistair Cockburn. Presented at the 4th International Multi-Conference on Systems, Cybernetics and Informatics, Orlando, Florida, June, 2000. Available online at <http://alistair.cockburn.us/crystal/articles/cpanfocisd/characterizingpeopleasnonlinear.html>