Software Developers: Architects or Artists?

1. SOFTWARE DEVELOPERS: ARCHITECTS OR ARTISTS?

Introduction

Part of my responsibility as managing director for a consulting company involves staffing development projects with programmers, database analysts and system administrators. For the most part, my interviewing techniques are standard.However, to end each interview, I ask candidates whether their work is more similar to that of an architect or an artist. Their initial reaction to this question is invariably the same. Candidates hesitate, trying to remember what they were taught about managing the software development life cycle or what they have read in the latest technology journal about rapid application development. More times than not, they give the answer that logically follows from years of exposure to academic and industry literature that attempts to formalize software development. Associating with architects is understandable considering that the analogy of construction in so ingrained in the software industry that terms like *system architecture, network engineer* and *stress testing* dominate today's IT jargon.However, I find it intriguing when candidates consider themselves more akin to artists than architects. It is interesting to hear how their work is not the calculating construction that it appears to be, but rather a less scientific process of creative experimentation. They derive much satisfaction from not knowing exactly how their projects will evolve until they are fully immersed in their technologies. Even though effective IT professionals must possess both the analytical skills of an architect and the creativity of an artist, exploring the artist metaphor provides valuable insight to many of the characteristics of successful IT development projects.

Traditional Software Development

The traditional software development process dictates that IT professionals develop products from predefined specifications with success defined in terms of time, cost and quality. A large portion of the process is devoted to mitigating the risk associated with information technology. Thorough specifications are developed before any programming begins to reduce the chance that developers misinterpret the project requirements.Formal system designs are constructed to ensure that the program performance, reliability and scalability are within acceptable limits. Lastly, comprehensive unit, system and integration testing is required before any program can be certified for production use.Many methodologies have evolved to facilitate this process, however most share these same basic components.

Creative Experimentation

Contrary to followers of the traditional software development process, candidates that consider themselves more like artists view software development as an evolving process. To them, projects consist of continually molding the database, user interface design, network and server technologies until the optimal product emerges. They often use a failed attempt in one area of the system to inspire improvement in another area.Just as an artist might discover a pleasant color scheme in his painting only after inadvertently applying too much contrast color, artistic developers often uncover new approaches to implement a program only after they start building the software solution. For example, in one interview, a developer explained how on a recent project her inability to improve database performance lead to moving server processes to the client, which unexpectedly expanded the capabilities of the user interface. Creative experimenters realize that most of the innovation in a software project cannot be planned. Consequently, these developers work best in environments that readily support modifications to design specifications after development begins.

Although the overall goal of any software development project is to create as much value as possible, it is virtually impossible quantify how much value can be created before the project begins. It is just as hard to estimate the business benefit of designing an internet-based customer service application as it is to predict the aesthetic value created by painting an oil based landscape. Both have the potential to create significant value, however, until the work begins no one will know whether you are producing a masterpiece headed for the Louve or a second-rate piece destined for the living room. Much of the value created through IT projects is not formulated during the design stage, but rather is uncovered during development as programmers craft their components. As a result, I find ideal developers are those who are apt at meshing various technical components together and have trained eyes to identify opportunities that expand the capabilities of their projects.

Conclusion

Obviously, no software development project can begin without some semblance of preparatory analysis and planning. Organizations must evaluate the high order of magnitude returns for each IT project investment through a basic understanding

Software Developers: Architects or Artists?

of the business problem that the software is intended to solve. They must also realistically assess its capacity to build IT systems to assess the inherent risk of time and cost overruns. Lastly, just as artists must decide which in medium to express themselves, whether canvas, clay or clarinet, companies must decide whether their IT project will be suited better by an internet, client-server or mainframe implementation. However, with a team possessing the aptitude for creative experimentation, it is not always necessary to extend this preparatory analysis much further than simply verifying the feasibility of successful implementation. If you are fortunate enough to be working with such a team, it is best to quickly begin experimenting with various means of implementation and allow the IT team members the flexibility to discover best solution.