**SOFTWARE DESIGN CHALLENGES**

Today, the software design phase has evolved from an ad hoc and sometimes overlooked phase to an essential phase of the development life cycle. Furthermore, the increasing complexity of today’s systems has created a set of particular challenges that makes it hard for software engineers to meet the continuous customer demand for higher software quality. These challenges have prompted software engineers to pay closer attention to the design process to better understand, apply, and promulgate well known design principles, processes, and professional practices to overcome these challenges. Some of the major challenges include requirements volatility, design process, quality issues (e.g., performance, usability, security), distributed software development, efficient allocation of human resources to development tasks, limited budgets, unreasonable expectations and schedules, fast-changing technology, and accurate transformation from software requirement to a software product.

A brief discussion of these challenges is presented next.

**Design Challenge 1: Requirements Volatility**

A major reason for the complexity of software projects is the constant change of requirements. When designed properly, software can be modified or extended easily; however, when designed poorly, modifying software can become overwhelming and lead to all sorts of complex problems. Unlike the development of computer hardware, bridges, houses, or mechanical parts, software’s very own nature allows itself to change to provide different or new functionality to systems. This same trait that makes software so desirable is what makes it also so complex. Although much effort is put into the requirements phase to ensure that requirements are complete and consistent, that is rarely the case; leaving the software design phase as the most influential one when it comes to minimizing the effects of new or changing requirements. Requirements volatility is challenging because they impact future or current development efforts. This forces designers to create designs that provide solutions to problems at a given state while also anticipating changes and accommodating them with minimal effort. This requires designers to have a strong understanding of the principles of software design and develop skills to manage complexity and change in software development.

**Design Challenge 2: Process**

Software engineering is a process-oriented field. Software processes allow engineers to organize the steps required to develop software solutions with schedule and cost constraints. Therefore, at the core of every software development company, there should be a sound, well-understood, and consistent process for software development. Processes can also be developed and customized for particular phases of the software engineering life cycle. In the design phase, software processes involve a broad set of activities and tasks that bridge the gap between requirements and construction while adhering to a set of project-specific (or company-specific) constraints. These activities include common ones, such as architectural and detailed design, as well as other supporting activities. These supporting activities include establishing a design review process, defining design quality evaluation criteria, evaluating design reuse, establishing design change management and version control procedures, adopting design tools, and allocating resources. In many cases, a company’s design process is not well established, is poorly understood, or is approached with minimalistic expectations that ignore aspects that are essential to executing a successful design phase. Focusing design efforts on creating independent software products, such as a simple class diagram or user interface, while ignoring other design activities may create complexities later on during system’s test and maintenance. The design process is challenging because essential design process activities are often overlooked, done in an ad hoc manner, or simply not done at all. In many cases, a well-established and well carried out design process serves an indication of future project’s success.

**Design Challenge 3: Technology**

Software is meant to be everywhere. From health-care systems and education to defense and everyday ubiquitous devices, software is required to operate on a massive and always evolving technology landscape. Besides the operating environment, the technology for designing and implementing today’s software systems continues to evolve to provide improved capabilities. Examples of these include modeling languages and tools, programming languages, development environments, design patterns, and design strategies. As new technologies emerge, software engineers are required to assimilate and employ them all at the same time. In some cases, emerging technologies do not completely replace old ones. Some software systems are required to interoperate with old legacy systems designed with older design methodologies. This results in software designers employing different design methodologies and technologies, all on the same software system. In other cases, design models need to be derived from existing code, modified, and made interoperable with newer technologies. This technology-driven aspect of the design phase creates a demand for capable software designers that can assimilate new technology quickly and effectively to succeed at designing software. The technology aspect of software design is challenging because it is fast and ever-changing; therefore, designers must keep abreast of the latest advances and become proficient in the application of these advancements while maintaining rooted in legacy technology.

**Design Challenge 4: Ethical and Professional Practices**

Designers create blueprints that drive the construction of the software. During this creation process, designers are required to determine how design decisions affect the environment and the people that use the software. In many cases, the software development process is traditionally carried out under tight schedule constraints. Inherently, all phases of the development life cycle suffer from this, including the design phase. This creates external pressures that can lead designers to deviate from the normal design approach to meet these demands, which can have catastrophic consequences. No matter how tight deadlines are, how much animosity exists within the design team, or how much other external/personal factors are brought into the design phase, software designers must exhibit strong ethical and professional practices to ensure that the systems they build are of highest quality and that all design considerations are properly evaluated. In many cases, this requires designers to exert strong leadership skills to influence and negotiate with stakeholders, motivate the design team, and lead the design process to accomplish the project’s goals. Designers are also responsible for enforcing ethical guidelines during the design process; evaluating the social impacts of their designs in the public domain or in safety-critical systems; and to follow the appropriate professional practices to ensure success in the overall system. The ethical and professional practices aspect of software design is challenging because designers are constantly faced with numerous pressures from stakeholders that influence ­designers’ decisions, most of which have consequences of social, ethical, or professional nature.

**Design Challenge 5: Managing Design Influences**

Designs are shaped by many different influences from stakeholders, the development organization, and other factors. These influences can have cyclical effects between the system and its external influences, such that external factors affect the development of the system and the system affects its external factors (Bass, Clements, and Kazman 2003). Managing these influences is essential for maximizing the quality of systems and their related influence on future business opportunities. Of specific importance are design influences that come from the system stakeholders and its developing organization.