Practical software design for engineering application development

Ibrahim Fahdah & Bassam Izzuddin  
*Imperial College, UK*

James Way  
*Steel Construction Institute, UK*

**Keywords:** engineering applications, software design practices, software development

Software development is in general a complex task and requires high skills and extensive experience. Many software projects are delivered late, over budget, and/or with less required features and functions.

This paper highlights some practical techniques for engineering design application development, which are critical to produce maintainable software design. The practices introduced in the paper are the result of long experience working with software development for engineering design applications. They have been recently employed and enhanced during the reconstruction of a well-known software for cellular beam design known as Cellbeam.

Technical functionalities required in an engineering design application can often be broken into two groups: generic functionalities and product-specific functionalities. Encapsulating the generic functionalities in a framework not only increases reusability but it also increases productivity and efficiency. Isolating the generic functionalities allows their implementation and testing to start whilst the rest of the system is being designed.

Components need to communicate to trade functionalities. There are different methods to facilitate communication between components in software. Communication requires coupling. Strong coupling is often used by developers because features are coded faster, direct relationships between components are easier to conceptualize, and problems are tracked down more quickly. However, Components with strong coupling are difficult to maintain as the software grows. Loose coupling, on the other hand, reduces the interdependencies between components which increases the system testability and maintainability. Loose coupling can be achieved using either an interface or messaging approach.

Each component in a software can be seen as a subsystem. The internal design for the system components usually evolves with time as the development progresses. There are some aspects found to be very critical when designing the technical components for engineering design applications. Interdependency and performance were at the top of the list. The former is considered important for maintainability and testability and the latter for efficiency.

It is advisable to reduce interdependency between modules within the same component. However, it was found that the concept is also practical to be applied on the method and function levels too. Self-contained functions are easier to test especially when the container module is subject to frequent
changes. This increases maintainability and testability of the component. Another good design practice which emerged from the reengineering process of Cellbeam is the separation between the input and output classes. The separation between the two types of data helped decrease dependency between the overall system components.

The performance factor is usually overlooked by some developers or left until it is too late to resolve. It was found that the calculation speed can be improved with some little considerations without even changing the logic (e.g. calculation algorithms). Performance considerations should target functions that are called a relatively large number of times or are the most computationally demanding. One additional practice that can sometimes improve the system performance is using a global cache to store any temporary output that is not part of the final desired results. This can reduce the need to keep calling and executing the code for time-consuming functions.

System testing is the task that is usually underestimated and overlooked to some extent. It is very important in modern software development to consider the testing plan before even the system implementation starts. Testing can sometimes influence the system and the component internal design. Keeping the component coupling loose increases the overall system testability. There are many types of software testing that are comprehensively described in the related literature. Among these types, regression testing is believed to be the most critical one when it comes to engineering design applications. Automated regression testing established in the early stage is very useful to test the technical components as the implementation progresses or very soon after each modification or error fixing. Without this routine testing, it would be almost impossible to have a flexible and maintainable system.

Modern software development is considered to be one of the most complex and challenging tasks. To develop software without a blueprint or set of overriding principles makes the task nearly impossible. With a well-defined guidance, large-scale software development can be a much more satisfying experience for all those involved. In this paper, we provided some practices and techniques which have proven, besides the common design patterns, to be valuable in several large and medium software projects in the field of structural engineering design applications.

References

CELLBEAM SOFTWARE, from http://www.asdwestok.co.uk and http://www.steel-sci.org
GAMMA, E., HELM, R., JOHNSON, R.E., VLISSIDES, J.M., 1994, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.
MICROSOFT, http://www.microsoft.com/NET