SOFTWARE DESIGN ISSUES- INFORMATION SYSTEMS ANALYSIS

Dr. Vijay Kumar Sharma

Devraj Group of Institutions, Ferozepur, Punjab, India.

Introduction

The final aim of research in software designing is to improve the quality and productivity associated with both the products and processes of software development. The foreseen benefits are improvements in the organization of the development process, increases in user's satisfaction, and the delivery of superior functionality to the market place in the structure of software and related products. A number of software designing paradigms, approaches and methodologies be present today. Although the object-oriented (OO) paradigm still remains one of most popular, it is gradually replaced by the aspect-oriented (AO) because of the concern crosscutting problem. Some of the major problems associated with the automation of software development occur with respect to requirements specification, design, reusability, maintenance, validation, verification, and testing.

Software Design/Development Issues - this mainly describes some of the issues associated with the software development process. Here in each case we define the issues, find the sources of its reality and discuss the implications of what to do timely. And then we see these issues as highly interdependent with other and observe the approach to an overall solution.

A. System Evolution Currently, the process of maintenance is very similar to that of patching of software bugs. The software maintainability is assumed wrongly to be a natural consequence of good systems development. These perceptions are faulty and thus results to the maintenance crisis being experienced today. So, now in order to support the scope of maintenance, it must be recognized that software systems need to evolve and that this evolution of systems cannot be treated in the same manner as initial development. Information systems are normally an important arm for the business and it must adapt continuously to the changing business environment. Change is not the exception for these software systems, it is just the model. Unless they can effectively adapt to the needs of the company, the competitive position of the company and the stability will be jeopardized. So in order to support system evolution, information required during maintenance must be available and processes which directly support maintenance must be developed. Taken together, design basis and the maintenance processes must help the designer - a) Understand the structure, functionality and behavior of a system (b) Review the level of impact of alternative changes, and (c) Control of the impact of necessary changes. Of course, maintainability and the integrity of the system must be preserved. Supporting system evolution must be viewed as a major factor in shaping an approach to formalizing the design process. Maintenance requires more information because an existing system represents a large set of constraints which does not exist during initial design and the design tradeoffs made during maintenance differ from those in the initial design process. For example, data structure decisions may be driven by efficiency and quality during initial development but by minimization of impact during maintenance.

B. Expertise and Experience Necessarily large size of development team members combine with the relative scarcity of highly experienced builders mandate the judicious use of less experienced personnel in most of the phases of development. Expertise and experience is concerned with the ability to successfully complete and manage large software development projects with a relatively small percent of highly experienced and proven resources. This is the critical issue given that the success of a project development is directly correlated with the levels of experience and talent across members of the project team. The main focus of research should be on supporting the high level design activities where the dividing of high level problems into relatively independent components takes place. The simple functional demands of individual programs allows for their construction by less experienced members.

C. The Design Process The design process is the task of generating or creating design by-product and subsequently refining, evaluating, integrating, and modifying these until the final result satisfies the initial requirements that are taken according to the problem definition. In core, the design process is the task of mapping problem requirements to design solutions. The design process should be guided by an economic, productive and controllable methodology that will ensure a high quality product. The reasons why the design problem is important are necessary. Good design decisions made early have a positive effect on the efficiency of the development process and the quality of the ultimate product. Poor design decisions declines the efficiency, quality and cost of the development process and the design products. Design of the software development is categorized by a necessity to deal simultaneously with a large number of diverse constraints. In general the design problems are intractable. Mainly our intention is to exploit the natural structure in ways that allow us to reduce the complexity of the problem to manageable levels. We mention several issues that help focus some of the design process concerns.

i) The Paradigm Problem: The paradigm problem mentions to the failure to develop and recognize a productive, manageable, economically feasible process model for soft designing. Attention has been focused on the development of new software designing paradigms, but till now no results have proven completely satisfactory for development. Any successful model must deal with the interdependent facts of making design decisions while recognizing the need for adequate and project management.

ii) Design Evaluation: So in order to make good design techniques and decisions, one must have the ability to assess the quality and validity of a particular design decision or can able to weigh the relative merits of competing design alternatives. The lack of evaluation ability definitely leads to inadequacies in assessing the impacts of a design decision on all factors of the design process. Under the umbrella title of evaluation we include testing, verification, validation and, as a specific instance, prototyping.

iii) Bridging the Functional to Technical space: A remarkable portion of the design process occurs when translating the business problem description in to a high level systems design. Today, the techniques at this level of the development process are not clearly understood. The result is we cannot able to map problem requirements to technical solutions. No good languages have been developed in which the requirements of the problem can be expressed and ultimately transformed to technical solutions.

iv) The Representation Problem: The representation problem is a elementary requirement for advancement in each of the areas mentioned above. The issue is the ability to manipulate, express and make inferences about design decisions and processes. Currently, major development takes place at very low level design for at least two reasons. Firstly, current methods of software engineering encourages the designers to think in terms of low level issues such as data structures, performance measures, database, screens, and interfaces etc because these representations are the only mechanisms that provide evaluation feedback and feasibility measures. So as a result designers move to lower levels without adequately investigating alternative early design decisions. Second, the nature of the business is that cost pressures often do not allow for a need to investigate on high level design alternatives. As a result, projects designs are often inadequate and easily damaged.

v) Large Scale Integration: The major problems of large scale integration is concerned with the understanding, usage and exploitation of the functionalities, protocols, standards and communication interfaces of a heterogeneous set of technologies in developing large systems. Integration of different component systems differing both in functionality and platform is important because systems within the commercial environment can no longer be treated as isolated entities. In fact there is great discourage to do so. The large scale integration problem mainly points out the need to understand the interrelationships of all systems within a company. Efforts need to be concentrated on large scale design at the enterprise, or companywide level before detail design and implementation of a particular component is undertaken. Understanding the enterprise level connectivity and integration issues is extremely important and will have tremendous impact upon the design process.

Conclusions: The intent of this paper is to discuss the suggestions that the process of developing very large information systems has on the approach to the software designing problem. It is our position that possibilities for enhancing the software development process are functions of the domain in which one participates. The paper proposes a classification of the ways of solving design problems using OO and AO design patterns. The proposed classification contributes to the better understanding of relations among the design problems and the design patterns. The paper proposes also a technique for redesigning object-oriented patterns into pure aspect-oriented ones.

REFERENCES

1. Craig Gaskell and Armstrong A. Takang,Professional issues in software engineering the werswective-of UK academics
2. [BOOK] Systems analysis and design methods LD Bentley, KC Dittman, JL Whitten - 2000
3. David NotkinAutumn Quarter 2008 Software engineering issues
4. Richard H. THAYER, ARTHUR B. PYSTER AND ROGER C. WOOD, Major Issues in Software Engineering Project Management.

5. Synthesis aspects of the Paradise design environment F.J. Rammig