Concurrent Engineering: A Roadmap to Software Engineering, Database, and Data Warehouse Technology

Shahriar Movafaghi, Southern New Hampshire University, USA; E-mail: s.movafaghi@snhu.edu
Hassan Pournaghshband, Southen Polytechnic State University, USA; E-mail: hpournag@spsu.edu
J. Stephanie Collins, Southern New Hampshire University, USA

ABSTRACT

Software engineering, database and data warehouse technology are closely related to each other despite being separate disciplines. In this paper we discuss how software engineering, database and data warehouse technology are inter-related and gain benefits from each other. We also examine the critical technical and managerial issues required to enhance potential benefits used in structure and/or object-oriented paradigms. We present recommendations and guidelines for applying Concurrent Engineering (CE) principles to software engineering, database and data warehouse technology.

1. INTRODUCTION

Engineering is defined as [1]: The *profession* in which a knowledge of the *mathematical* and *natural sciences* gained by study, experience, and practice is *applied with judgment* to develop ways to *utilize*, economically, the *materials and forces of nature for the benefit of mankind.* Software (including the user interface), database, and data warehouse engineering are by definition a kind of engineering, and therefore carry the same set of social responsibilities as all of the other kinds of engineering [2]. Common engineering processes, such as requirement analyses, design, implementation, testing and maintenance are applied to software, database, and data warehouse engineering in the same way as other branches of engineering including civil, electrical or chemical engineering. However, software, database, data warehouse engineering are subject to particular frequent changes, including those imposed while a product is under development [2].

Software, database, and data warehouse engineering are different disciplines with common related issues and processes. We examine the differences among these disciplines and relationships and provide some of the potential benefits. This paper discusses how concurrent engineering is applied to software, database, and data warehouse technology. Comparisons of software and database technology have been made in the literature [3, 4]. In this paper we extend the research to 1) data warehouse engineering; 2) uses of structure and/or object-oriented paradigms; and 3) the role of concurrent engineering.

2. RELATED DISCIPLINE AMONG SOFTWARE, DATABASE, AND DATA WAREHOUSE ENGINEERING

Various methodologies and principles are proposed for software, database, and data warehouse technology [2, 5, and 6]. However, they are related in many aspects as mentioned below:

a. Project Planning – All three disciplines are required to do the project planning. A *SWOT analysis* that examines a company's strength (S), weaknesses (W), opportunities (O), and threats (T) can be used in software project analyses. The questions asked often lead to information technology related issues, which in turn requires further review, analysis, and planning [7]. Other techniques such as 1) understanding the problem or opportunities, 2) defining the project scope and constraints, 3) performance of fact-finding, 4) evaluating feasibility-operational, technical, economic and schedule feasibilities, 5) estimating project development time and cost, and 6) readiness "Litmus test" [6], can apply to all disciplines. Front end and risk reduction analysis is usually applied to large projects and are also applicable to all three disciplines.
b. Requirement Analysis – Techniques to obtain requirements such as interview, questionnaire, sampling (systematic, stratified, or random) as well as modeling (such as the use of a UML use case diagram or activity diagram) is similar in software, database, and data warehouse technology. Note that in many instances, a database or data warehouse requirement starts when the software engineering requirements are finished. We discuss this issue in more detail in section 3 with recommendations of concurrent requirements for all disciplines involved.
c. Design – If object-oriented methodology is used, then the class diagram is a *superset* of the E-R diagram or dimensional diagram. Therefore, the class diagram should be used for both the logical and physical model of a database or data warehouse. The database and/or data warehouse modeler should become familiar with the class diagram and not use an E-R diagram and/or dimensional diagram. If there are design conflicts between the disciplines involved, then concurrent engineering techniques should be applied to resolve the issues as discussed in section 3.
d. Implementation – There should be a single coding, database and data warehouse standard that can be shared among these disciplines.
e. Testing – Use case scenarios can be used as testing scenarios in each of the software engineering, database and data warehouse technology disciplines. Other methods used in unit, integration, system, acceptance and regression test can also be applied in all disciplines
f. Maintenance – Procedures and guidelines (such as: traceability) used to maintain a software engineering project can apply to database or data warehouse technology projects.
g. Tools – There are various tools that cover different spectrums of SDLC. The selection of CASE tools is discussed in the literature [8, 9]. However, one should attempt to obtain a CASE tool that can be used in a multi-discipline environment.
h. Versioning and Configuration Management – Appropriate versioning techniques and metrics are required to map different versions of the software, data base schema and data warehouse schema. The techniques and tools for configuration management can be shares among these disciplines.

3. CONCURRENT ENGINEERING

There are many definitions of CE such as [10, 11 and 12]: systematic approach to integrated and concurrent development of a product and its related processes. Concurrent engineering emphasizes a response to customer expectations and embodies team values of cooperation, trust, and shared-decision making. It proceeds with large intervals of parallel work from all lifecycle perspectives, and is synchronized by comparatively brief exchanges to produce consensus. Concurrent Engineering can be defined as the integration of interrelated functions at the outset of the development process in order to minimize risk and reduce effort down-stream in the process, and to better meet customers' needs. Multifunctional teams, concurrency of product/process development, integration tools, information technologies, and process coordination are among the elements that enable CE to improve performance.

Software systems architecture has four basic components, namely: 1) *presentation* component which is a system interface with the user such as a web, graphical user interface, or voice response unit. The user interface engineer works on this component. 2) *process* component which defines what software component (or objects in the case of object-oriented systems) to be processed by the business rule component. A software engineer works on this component. 3) *business rule* component processes the business rules of the organization dictated by a process component. Business rules are usually dynamic and implemented by a rule engine. Software engineers work on this component. 4) *data access* component retrieves, inserts, updates, or deletes data from the database and/or data warehouse. All SQL (Structure Query Language) commands are written in this layer if a relational database is chosen for the system. Database and data warehouse engineers work on this component. Database and data warehouse engineers are also responsible for the logical and physical design of a database and/or data warehouse. Normalization techniques (usually a third normal form) is used to avoid redundancy in the database design. A *dimensional* model is used in a data warehouse that allows redundant data. Figure 1 shows typical software components. This process is *sequential:* the activities from one activity are handed off to the next only after completion. There is little or no cross-communication among various functions [13].

Figure 2 shows concurrent engineering applied to in software, database and data warehouse engineering. This process has extensive overlap and the team structure is multifunctional. For example, when a UI engineer is working on the design of a screen, the database engineer reviews and gives its feed back to UI engineer to ensure appropriate criteria is met. That may include information such as, if the implementation of the screen requires a nine way join between the underlying tables, then the database engineer recommends the screen be changed to one that requires a smaller number of joins (like two or three) to meet the performance criteria. Although concurrent engineering requires challenging coordination, it reduces the time to market, major iterations and increases overall project performance. Concurrent engineering requires extensive use of information technology tools including coordination tools.

The recommended guidelines for concurrent engineering are [13]: 1) define and formalize the CE process. 2) define all overlapping of activities. 3) identify process ownership. 4) set clear, quantitative goals.

Figure 1. Software system components

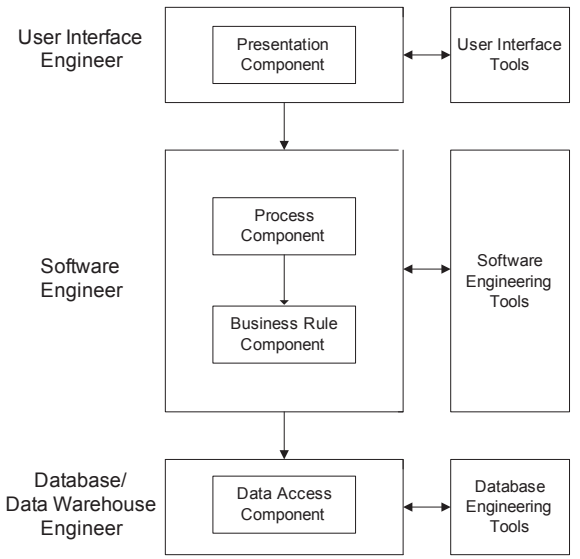
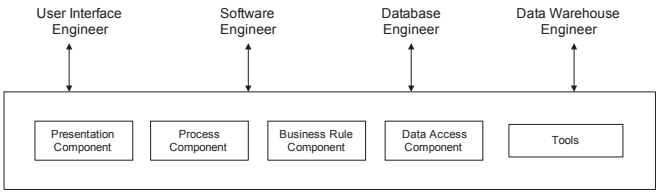


Figure 2. Concurrent engineering: Software, database and data warehouse engineering



4. CONCLUSION

Software engineering, database, and data warehouse technology are three different disciplines. However, these technologies are in many ways related to each other and can benefit from each other. We survey what has been done regarding this issue, and discuss what else can be done to further clarify the roadmap. Concurrent engineering is widely used in manufacturing. The application of concurrent engineering principles in software engineering, database, and data warehouse technology were discussed. Further research is required to obtain numerical data (such as time to market, reliability, and maintainability) for projects that are implemented using concurrent engineering verses traditional engineering methods.

REFERENCES

1. Accreditation Board for Engineering and Technology (ABET), 1996.
2. Eric J. Braude, "Software Engineering: An Object-Oriented Perspective", John Wiley & Sons, 2001.
3. Hassan Pournaghshband and Shahriar Movafaghi, "Software Engineering and Database Technology", The Association of Management / International Association of Management Conference, 2005.
4. Klaus R. Dittrich, et. al., "Databases in Software Engineering: A Roadmap," Proceedings of the Conference on the Future of Software Engineering, Ireland, 2000.
5. Ramez Elmasri, et. al., "Fundamentals of Database Systems," Addison-Wesley, 2000.
6. Ralph Kimball, Laura Reeves, Margy Ross, and Warren Thornthwaite, "The Data Warehouse Lifecycle Toolkit, Expert Methods for Designing, Developing, and Deploying Data Warehouses", John Wiley, 1998.
7. Shelly, Cashman, and Rosenblatt, "Systems Analysis and Design", Course Technology Co. 2006, 6th Edition.
8. Hassan Pournaghshband and Shahriar Movafaghi, "A Practical Approach to the Evaluation and Selection of CASE Tools", International Conference on Software Engineering Research and Practice (SERP'03: June 23-26, 2003, Las Vegas, Nevada, USA).
9. Hassan Pournaghshband and Shahriar Movafaghi, "Evaluation and Selection of CASE Tools for Educational Purposes", IRMA International Conference, May 15-18, San Diego, California, USA.
10. www.sei.cmu.edu/opensystems/glossary.html
11. Winner, R.I.J.P. Pennel H. E. Bertrand and M. M. G. Slusarezuk, 1988. The Role of Concurrent Engineering in Weapons System Acquisition. The Role of Concurrent Engineering in Weapons System Acquisition. Institute for Defense Analyses, Alexandria, VA, U.S.A. IDA Report R-338.
12. Bessant J. 1991, New product Development: The New Time Wars, in J. Blackburn (ed.). Time-Based Competition: The Next Battleground in America Manufacturing. Homewood: Business One Irwin.
13. Bhuiyan, N., Thomson, V. and Gerwin D., "Implementing Concurrent Engineering", January-February 2006, Research – Technology Management.