To: Kurt Wallnau, kcw@sei.cmu.edu, ICSE98 CBSE Workshop. Position Paper Submission 2/20/98.

'A Validated Software Reuse Reference Model Supporting Component-Based Management'

Authors: Nader Nada, PhD (point of contact) nnada@osf1.gmu.edu; David C. Rine, Professor drine@cs.gmu.edu. Department of Computer Science, School of Information Technology and Engineering George Mason University, MS 4A5, Fairfax, Virginia, USA 22030-4444

This paper reports on a successful Components-Based Development (CBD) approach that uses a well-defined, empirically validated software Reuse Reference Model(RRM). None of the currently existing models have considered a comprehensive software business perspective (non-technical activities) based upon reuse as an approach to design their effort and estimation models. If this developmental problem can be solved, progress will be made towards a systematic approach to software reuse. Our thesis is that CBD based upon a validated software reuse reference model proves to be a practical solution to decreased software development effort, increased software product quality, and decreased time-to-market, especially if it is applied in a systematic way across the software development life cycle, including reuse of requirements, specifications, designs, documents, codes and verification/validation plans.

The critical problem in today's practice of software reuse is a failure to conceptualize, define and develop necessary details to support a valid software reuse process reference model. The software development industry will not be successful in utilizing and managing CBD paradigms until there is a conceptualized, well-defined and "validated empirical reference model" for software manufacturing that incorporates best software reuse practice and that can be customized for different kinds of software development enterprises. In the research reported in this paper a theoretical model of the software reuse reference model is presented, based upon prior knowledge and expertise about software reuse. Then, this model is empirically validated using three different empirical studies. The definition and validation of this model has been carried out using four steps. First, the reference model is developed based on prior work. Second, this theoretical reference model is empirically validated using both case studies and surveys. Third, the impact of the reference model on software development effort, quality, and time-to-market is empirically derived. Fourth, an initial set of successful cases based upon the software reuse reference model utilization are identified. The main contribution of this paper is a well-defined and validated reference model for the practice of software reuse for CBD. A secondary contribution is an initial set of case studies from software development enterprises having both a high success from the practice of reuse in terms of decreased effort and increased quality and a high correlation in their application of our software reuse reference model features.

One of the keys to CBDs success is a standard infrastructure for components. Infrastructures need three main elements [Kiely 98]. First, a uniform design notation is needed that provides a standard way of describing components functions and properties, which would be critical to designing collaboration between components. The industry is rapidly adopting the Uniform Modeling Language (UML), which combines several design notation, as default standard. Second, repositories are needed as a means of cataloging available components with a description of their features would let developer find the components appropriate for an application. The Object Management Group (OMG) is developing a repository specification, whose core will be the Microsoft Repository, currently the dominant specification. Third, a standardized CBD interface is needed that lets any application in any language access components features by, for example, binding to the component model or interface definition language. The OMG is trying to make both Microsoft's Distributed Component Object Model (DCOM) and Sun Microsystem's JavaBeans specifications interoperable with its final Common Object Request Broker Architecture (CORBA) specification.

Our study investigates the success of our software Reuse Reference Model (RRM) in predicting effort, quality, and time-to-market as well as explores additional facets of cost behavior. Process measures are defined and collected, and the use of our model as an assessment tool throughout development is tested.

We expanded Rine's model [Rine-Sonneman 97] and have incorporated additional important technical and non-technical activities such as: Reuse Quality Characteristics, Quality Control, Best Development Practices Experience , Software Standards (CORBA, DCOM, etc.), Trends of COTS, Case-Tools, Training, Market Place, Financing Marketing Forecast, and Reuse Metrics and Effort Estimation Models.

From the case studies, survey, and QSM study results Nada [Nada 97] concluded that the software reuse reference model implementation level (RRML) is a predictor of decreased software development effort and decreased time-to-market.

A. The Survey

1. Problem Statement

"There does not exist an effective and validated CBD RRM which incorporate both the technical and non-technical facets of software reuse and showing its impact on software development effort, quality, and time to market.

2. Empirical Research Validates the Model (Survey Results)

The survey examined 34 questions about the software manufacturing RRM that incorporates both the technical and non-technical activities that might be needed to establish a successful software reuse program in the organization. The survey data analysis answered the following questions (Q1-Q5):

Q1. Is the software RRM implementation Level (RRML) a predictor of:

a. products development effort
b. time-to-market
c. product quality?

A1. Yes, there is a correlation between the RRML and the organization improvement level of their software development effort, time-to-market, and quality. (SQ23-SQ26)

Q2. Is the reuse percentage a predictor of the RRML?

A2. No, there is no correlation between the reuse percentage and the RRML?

Q3: At which phase of the software life cycle do organizations get the greatest benefits from software reuse?

A3: By considering the reuse approach during the early phases of the software development life cycle. (SQ27-SQ29)

- 57% of the projects realized high commonality with the requirements of previous project(s) at requirements phase.
- 43% of the projects realized high commonality with the design of previous project(s) at design phase.
- 38% of the projects realized high commonality with the code (documentation) of previous project(s).

Q4. How much do projects collect and utilize reuse metrics and use effort estimation models with reuse as part of their software development life cycle?

A4. There is a lack of collecting reuse metrics and using effort estimation models with reuse.

24% of the projects measure and analyze the software reuse process carefully to identify weaknesses, and have plans established to address these weaknesses.

Q5: Is the reuse process common practice and an integrated part of the organizations software development process?

A5. No, 33% of the projects developers and maintainers precisely follow a software reuse process which is defined and integrated with the organization's software development process.

B. The Case Studies

1. Introduction

Software reuse is incorporated by establishing a RRM. The RRM would contain those activities that are common to software development organizations whose software reuse success is high. Incorporation of such a software RRM would promise four things:

1. Decreased products development effort;
2. Increased product quality;
3. Potential decreased in time of product to market;
4. Overhead in establishing and maintaining the RRM that would eventually be offset by 1. through 3. above.

Companies today are faced with new and more challenging market pressures. In response, they have to reduce the time-to-market with new or enhanced products, increase the diversity of products available to the customers, and enhance the standardization and interoperability of the products. Reuse is a means to achieve such objectives.

Drawing from the results of these applications as well as analyzing the reports of experiences in the same field available in the literature, this research addresses the challenge of providing managers with qualitative and quantitative indications about the potential benefits of software development with reuse.

That means we looked closer at reuse programs in several organizations and analyzed the dimensions of software reuse in them . We tried to identify the following:

- The reuse level (%)
- The decreased level of development effort (or increased productivity)
- The increased level of product quality
- The decreased level of development time
- The decreased level of time-to-market

Finally, the Software Reuse experiences concerning 27 different organizations is reported in the form of case studies to let the reader into the detailed description of their background, application contexts, organization software reuse peculiarities, technologies, metrics, achievements as well as some specific problems and adopted technical solutions.

2. Software Reuse Case Studies Strategy

Many software development organizations believe that investing in software reuse will improve their product and process productivity and quality, and are in the process of planning for or developing a software reuse capability. Unfortunately, there is still little data available on the state-of-the-art-practice of investing in software

reuse. A 1991 study of Japanese Software Factories stated that in 1991 the Japanese software factories were the only organizations successfully applying software reuse [McCain 91]. This 1991 study and a second 1992 study of U.S. software developers covering 29 organizations [Frakes-Fox 95], are, until now, the only few major studies involving a large number of organizations. The majority of current information available on software reuse investment comes from the literature, which contains unproved theories or theory applied in a limited way to a few pilot projects or case studies. [Rine-Sonneman 97]

There are also some independent research studies such as:

- Software Reuse Using Frame Technology, by QSM Associates, Inc. 1994
- Software Evolution & Reuse (SER), The SER Experience Book, 1996

3. Case Studies Research Method

Only a thorough analysis of the specific organizational, managerial and technological context pertaining to the company can lead to the appropriate definition of a competitive company strategy and to the choice of the best suited approach. Nevertheless, the experience collected in the case studies offers useful hints as to which different conditions lead to the selection of different strategies, which are the most common problems encountered and which are the subsequent corrective actions to be taken.

It is our believe, though, that there exists general software RRM which establishes software reuse guidelines that were effective in our experience through these case studies and that could trigger and help new experiments.

C. Summary

The objectives of this research are: 1. the developing of a theoretical model of software RRM; 2. empirically validating the model; 3. providing the empirical impact of the software RRM implementation level on the software effort, quality, and time-to-market.

For validating the theoretical model we used three different research approaches:

1. Carrying out 27 case studies of industry and government software developers;
2. Surveying the industry and government software developers (93 projects);
3. Carrying out Quality Software Management (QSM) study of 19 selected projects.

The organizations used in the empirical validation of the RRM with the highest software reuse capability measures use the following components: product-lines, architectures which standardize product interfaces and data formats, common software architecture across the product-line, design for manufacturing, domain engineering, reuse process, management which understands reuse issues, software reuse advocate(s) in senior management, state-of-the-art reuse tools and methods, precedence of reusing high level software artifacts such as requirements and design versus just code reuse, and trace end-user requirements to the components (systems, subsystems, data sets, algorithms and/or software modules) which support them.

Therefore, software reuse solutions take two major steps. First, the RRM is integrated into the engineering and business functions of CBD. Second, software domains and product-lines are instantiated to the RRM. Another result is a RRM incorporating the extended international Object Management Group (OMG) UML/CORBA/DCOM. This specific development RRM, will itself be a reusable RRM from which one starts when developing what has previously been coined 'generic software tool kits' of various kinds, useful software industry.

From the case studies, survey, and QSM study results Nada [Nada 97] concluded that the software reuse reference model implementation level (RRML) is a predictor of decreased software development effort and decreased time-to-market.

Some Supporting References.

Frakes, W., and Fox, C. (1996). Reuse failure modes. IEEE Transactions on Software Engineering , 22(4) April.

Frakes, W., and Fox, C. (1995). Sixteen questions about software reuse. Communications of the ACM, 38(6), 75-87 and 112, June.

Lim, W. (1994). Effects of reuse on quality, productivity, and economics. IEEE Software September, 11(5), 23-30.

Nada, N. (1996). Progress Report for the Earth Science System Fellowship Program.

Nada, N. (1997). Software Reuse-Oriented Functional Framework, Ph.D. Dissertation, George Mason University.

Patterson, F. (Editor 1996). A NASA Approach to Reuse. Proceedings of the NASA International Workshop on Software Reuse, George Mason University, Fairfax, Virginia, USA, September.

Rine, D., and Sonneman, R. (1997). Investments in reusable software: a study of software reuse investment success factors. The Journal of Systems and Software. 1-18, Spring.

Software Productivity Consortium (SPC). (1993). Reuse Adoption Guidebook, SPC-92051-CMC, Ver 02.00.05, Software Productivity Consortium, Herndon, Virginia, November .

Sonnemann, R. (1995) "Exploratory Study of Software Reuse Success Factors", Ph.D. Dissertation, George Mason University.