CS551 – Paper Critique
Student Name - Dhananjay Indurkar Email ID: di5x7@umkc.edu

Software Reuse Strategies and Component Markets

By T. Ravichandran and Marcus A. Rothenberger.
Communications of the ACM August 2003/Vol.46 No. 8 Pp. 109-114

Summary of the paper

Today, software reuse is becoming increasingly important in improving software development efficiency and increasing the quality of software systems. People in the industry have known two general types of reuse strategies –**white-box** (when the source code of the component is available) and **black box** (source code is not available) reuse. The black box approach is to use software components "as is," with no code modification. It can further be classified into black-box reuse with in-house component development, and black-box reuse with components procured from the marketplace.

In recent years, black box reuse has become popular. This is attributable in part to the growing use of component-based development (CBD) technique to build software systems. The components and frameworks may be either developed in-house or commercially available off-the-shelf components could be used. Procuring components from the marketplace can increase the reuse rate. This approach allows developers to search from a potentially larger set of components, increasing their chances of identifying ones that fit their needs. It has been augmented by the recent standardization of reuse architectures such as CORBA, JavaBeans, and Microsoft's Component Object Model (COM) that enable software reuse across software development projects which was not possible in the past. Significant economic benefits by way of cost-reduction have been reported when using CBD approach with black box reuse. If a suitable black-box component is available in the repository, use of this component is preferable to other alternatives, as the configuration cost of a black box component is lower than the customization cost of a white-box component or its development from scratch.

The paper does a feature-by-feature comparison of the reuse strategies is done in a tabular format to illustrate the benefits and the problems associated with each type of reuse strategy. In choosing among the reuse strategies, organizations are confronted with a trade-off between component acquisition cost and component customization cost, and the different reuse rates achievable with each strategy. The authors identify the key parameters that determine customization and acquisition costs, and discuss an approach to assess these costs. They also note that component markets are still in their infancy and are supply-constrained. One implication of a supply-constrained market is higher component acquisition costs, which include the search cost and the component price.

Strength

The paper performs an in-depth examination of the different component reuse strategies. It analyzes the merits of the software reuse philosophy from the industry viewpoint by factoring in the cost and scalability factors.

Weakness

While the authors advocate increase in the use of third party off-the-shelf components, they do little to mention the efforts going on the industry to organize the component market and make hunting for the right components more systematic with increased chances of success.

Interesting Points

The authors state that the initial development cost is higher for black-box components as compared to the white-box components because of the need to parameterize the component so that the same component can be used in different configurations. And that this cost has to be amortized over several reuse instances. That being true indeed, there are other ways too in which the black-box component can pay off for its high development cost. As the black-box component exposes standard and well documented interfaces, maintenance becomes a lot easier for large and complex enterprise wide systems designed with such components. This way the black-box components offset their high initial development cost by helping achieve substantial savings in maintenance through the entire life of the application.

Critical Questions

1. What parameters to consider when determining which component based development strategy to use?

2. When choosing to use black-box components with in-house development strategy to build applications, what should be the critical mass of the component repository(number of components stored) to make this reuse strategy successful and still keep the repository manageable, as mostly the same team who develops the application is responsible for component development too.

3. Does the application architecture have enough granularity to allow extensive use of black box reuse strategy?

4. What level of support can be expected from third party component vendors when it comes to scaling the application?

5. Would the commercial off-the-shelf component market reach a level of maturity when it would become possible to easily replace/upgrade existing components in a business application with components from competing vendors that achieve the similar functionality by conforming to a rigid specification without having to do a lot of tweaking of the business application?