“Software Reuse Strategies and Component Markets”
Written by T.Ravichandran and Marcus A Rothenberger

Description: This paper starts with an analysis of the three strategies of software reuse: white-box reuse, black-box reuse with in-house component development, and black-box reuse with components procured from the marketplace. It discusses the advantages/disadvantages and applications of all of them.

It talks about the cost of reuse: customization cost and acquisition cost, how to estimate customization cost by analogy, and how to make a decision of whether use black-box reuse or white-box reuse by using the component-level decision tree.

It addresses how the market affects software reuse. “Black-box reuse with component markets could be the silver bullet solution that makes software reuse a reality, and advances software development to a robust industrial process—but only if market makers address the growing pains plaguing this immature industry.”

It points out that black-box reuse with component markets could be the silver bullet solution that makes software reuse a reality and as a consequence moves software development from a craft to a robust industrial process.

It also mentions several issues that need to be addressed—licensing, ownership and quality assurance of components. Unless the market makers gear up to address these issues, component markets could go the way of many other innovations in software development—a lot of hype and no discernable improvement in software development practices.

Strengths:

1. The idea of being able to reuse software architectures is intriguing. If an organization could reuse a proven architecture, it would save time and effort on projects which would lead to a competitive advantage because of the reduced time to market.

2. Although software reuse is not a new idea, it is still immature. The paper discusses how to improve software reuse—cost, quality and flexibility, etc., which will help the development of software reuse.

3. The paper compares the traditional development approaches and component-based approach. These comparisons highlight the importance of understanding the trade-off involved in using the various reuse strategies and judiciously choosing the one best suited for a specific development context.
Weaknesses:

1. Software reuse requires an up front investment in time and effort which can be difficult to fit into already tight schedules.

2. Reusing architecture (black-box reuse) and reusing codes (white-box reuse) has the same basic problem – there is a definite functionality/flexibility tradeoff. For architecture to be a useful reusable architecture, it must be flexible enough to work with a number of different projects as well as being “good”.

3. The acquisition cost (search cost and component price) could be very high at the present time due to the immaturity of the component market, which will limit the development of software reuse. We still have a long way to go.

4. The paper does not provide any specific statistical analysis to support its points.

Questions:

1. How similar must the projects to be in order to reuse architecture? How flexible the black-box should be? Is there any way to determine these?

2. Is there any quantitative way to determine the customization cost and search cost of software reuse?

3. How often is software reused (what is the productivity of the component)? How can the developer determine it?

References:

http://sern.ucalgary.ca/courses/seng/693/W98/couprie/reused.html
http://sern.ucalgary.ca/~cmcphee/SENG693/02-Dec-98.html