David Lambertson

**The Re-engineering and Reuse of Software Summary**

**Important Points from Article**

* “Software re-engineering and reuse are concerned with maximizing software usage for any given development effort. The production of software is expensive, and with the decrease in the cost of hardware and the increase in hardware capability, we have been led to ever more ambitious development projects, while qualified and experienced software development staff are in short supply.” [335]
* “It is the objective of software reverse engineering and reuse to recover some of this investment. We define the basic terms below, in preparation for the later fuller description of the areas in the body of the paper.” [336]
* “In order to make changes, we have to first understand the software, often involving 47 to 60 percent of the maintenance effort because of inadequate software documentation.” [336]
* “This complete cycle of reverse engineering followed by forward engineering is called re-engineering.” [336]
* However, reuse can also be applied much more broadly to include the redeployment of designs, ideas, or even the skills and experience of people.” [336]
* “Method and CASE suppliers will commonly include tools to revert the software to their own notations. Early tools here were for PSL/PSA, and more recently this has been done for dataflow diagrams (Excelerator) and data structures (Bachman).” [338]
* “Formal transformations have usually been viewed as tools of forward engineering, for example transforming a simple but inefficient algorithm into one that is equivalent but more efficient, such as replacing recursion by iteration.” [338]
* “There is also a need to consider more general knowledge about the area or domain of application of the components, as shown at the bottom of figure 4.” [342]
* “Each reusable component should be given a clear specification and description of the principles and concepts underlying the component, independent of any particular implementation.” [343]
* “A good system of classification not only provides a means for finding a particular piece but also provides a mean for finding a particular piece of software held in a library.” [344]
* “Exact matching of precise (formal) specifications will in general be undecidable (that is, impossible), and we therefore must necessarily reduce our descriptions for search purposes, accepting that we can only find near matches.” [344]
* “While many people claim that code reuse is irrelevant and that higher level design and specification reuse carries the benefits, frameworks actually help these ideas become concrete and become a special form of code reuse.” [ 344]
* Having selected a component (or set of components) for reuse, it is necessary to adapt the components for its intended use by composing it with other components and new software to achieve the desired results.” [344]
* “When specialized and connected together, the available code components will probably be insufficient to meet the full requirements and other original code may be necessary, perhaps to transform the output from one component to the required form of inputs to another.” [345]

**Things I Didn't Agree With**

“People who work in software production like producing software and will develop software rather than look for existing ideas, algorithms or code.” [346]

I don’t agree with this statement by the author. Software developers are lazy and will try to get something, whatever it may be, as quickly as possible. Using already built technologies and components will speed up any project a software developer may be working on. We may enjoy writing code but we also enjoy getting projects done quickly and efficiently.

**Things I Did Not Understand**

I understood the whole article.