Program Usability and Portability

Software development is a complex and complicated process that has several different aspects that affect the outcome of a product. A couple of aspects of the software development process that can force the process to become slow and difficult are creating algorithms and objects from scratch and creating a product that needs specialized hardware configurations. Creating algorithms from scratch takes time and usually introduces new faults into the program which forces a developer to spend more time debugging the program. A product that requires specialized hardware configurations can be difficult to maintain since the actual hardware can change every couple of years. A good software development practice is developing programs based on the properties of reusability and portability.

Reusability is the ability to use components of one product in the development of another product with different functionality. Reusing code or algorithms from previous products can greatly decrease the development time and the number of faults introduced into the system. The peg puzzle project reused the design of a board object with only a few modifications from a previous chess program to model the peg puzzle board. While the chess and peg games use two different sets of rules, both games have a board that is responsible for enforcing the rules of the game and therefore can be reused with minor modifications to the rules. While developing the peg puzzle solver, I tried to write my own backtracking algorithm from scratch instead of using an algorithm that somebody else had already wrote and probably debugged. The algorithm that I
wrote introduced a major fault when trying to solve the puzzle that would not have occurred if I reused the well known algorithm that other people had already written. Reusability is important when developing software because the developer does not waste time reinventing objects or algorithms which will in turn introduce less faults and shorten the development process. Parts of the peg puzzle project, such as the backtracking algorithm and the board object, can be easily modified and reused in future products.

While reusability can decrease the cost of product during the development and implementation, portability can help decrease the cost and increase the life span of a product through the maintenance phase of the product. Portability refers to the ability to inexpensively modify a product to run across different operating system configurations. This is an important practice for a company because it allows them to sell a product to various users and can allow them to not waste money in a couple of years making significant changes to the product due to new hardware. The text based version of the peg puzzle project that currently runs on Linux is fairly portable because it would not be expensive to make a Windows version and the program will not be significantly affected by different hardware configurations. The graphical version of the peg puzzle project is not portable because it is somewhat tricky to convert the program into a Windows version and the ability to perform the graphical animation relies on the hardware configuration of the computer. The ability for a company to produce a product that is portable can improve the success of the product and can potentially allow a company to spend less money on maintaining the product.

While developing the peg puzzle project, it was fairly clear how reusability was helpful but not so clear how portability could be used and helpful. The peg puzzle reused previously implemented object and algorithm that help reduce the amount of debugging required to test the
functionality of the program. While developing the peg puzzle program, the concept of developing software that is portable was not a high priority but is virtually important in the success of a large scale product that a company would produce for various users. The ability to successfully implement reusable and portable code is important because it costs the development and testing costs and allows the software product to be successful for many users over several years.