Managing the Evolution of Software Product Line

Cheng Thao, Michael Levi Haufe
Advisor: Ethan V. Munson

Software Product Line Engineering (SPLE) is a systematic approach to software reuse and for managing families of related products. Under SPLE, developers produce a set of core assets that are intended to be shared by multiple products. Each of the products has some its own product- specific code, which at minimum will be a bit of "glue" to connect a selection of core assets, and at maximum, may provide substantial additional functionality and dominate the shared code. All of this code, both core asset and product- specific, can evolve over time for the normal reasons of maintenance and evolving functionality.

Traditional SCM systems support single product development well. Software product lines pose a different evolution problem not solved by traditional configuration management systems because there are multiple distinct products that share a common code base that is evolving over time. Evolution in software product lines is viewed as a two dimensional problem consisting of time and space dimensions. The time dimension refers to the evolution of shared code and a product over time, while the space dimension refers to the variability among the products in the product line. In single product development, the evolution problem is a one dimensional problem: the time dimension. The problem of managing evolution of a software product line (SPL) is challenging because of the independent evolution of products, core assets, and their interactions. In this poster, we present an approach to capture the evolution of SPLs and core assets relationships, and to support change propagation and product derivation.

To do this, our prototype must model SPL, track shared core assets, represent how products are derived from the core assets, capture the independent evolution of the products and the core assets and allow changes to propagate between core assets and products. We use the product versioning model, which versions the state of a project as one item rather than versioning individual files as separate version items. A product line is modeled by a single core assets project and one or more product projects. Core assets are developed in the core assets project while each product is developed in its own product project. Each project has its own version space, so that they can evolve independently of each other. At a lower level, each project is composed of components that may or may not be shared. Our approach uses shared components to enable reuse by implementing sharing within the versioning system. Shared components provide an indirect reference to material stored elsewhere in the versioning system. A shared component can refer to objects of any granularity, from a single file to a directory containing files or other directories. Shared components are versioned items themselves, so that different versions of a product can include different versions of the same core asset, or even different core assets entirely.

We evaluate our prototype using a Graph Product Line (GPL), a standard problem for evaluating product line technologies. Since graphs are well understood data structures, they are a prime candidate for being a simplified, but realistic instance of a product line to evaluate software product engineering tools. It's used as the standard evaluation in SPLE research community. Our goals are determine if our prototype can model a realistic software product line, capture its evolution, and evaluate change propagation between core assets and products.

We have described an approach that is capable of versioning multiple types of product line projects including document project lines. It has a version model for a product line consisting of a single core assets project and multiple product projects where core assets are shared among the products through the use of shared components. Using the shared component data structure and the branching of the core assets project, we are able to support independent development of core assets and products and change propagation between them.