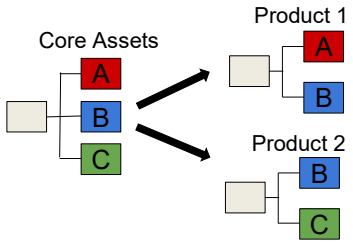
Managing the Evolution of Software Product Lines

Cheng Thao & Michael Haufe {chengt,mlhaufe}@uwm.edu
Department of Electrical Engineering & Computer Science
UNIVERSITY of WISCONSIN UWMILWAUKEE

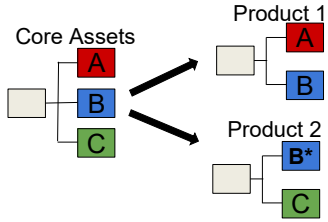
INTRODUCTION

Software Product Line Engineering (SPLE) is a systematic approach to software reuse that focuses on managing families of related products [1], [2]. Under SPLE, developers produce a set of core assets that are intended to be shared by multiple products. Each of the products has some of 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 code, can evolve over time for the normal reasons of maintenance and evolving functionality.

Software Product Line (SPL)



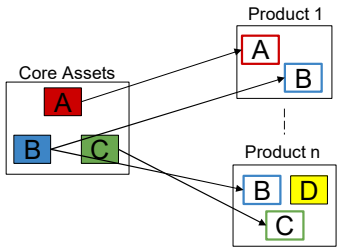
Evolution



APPROACH

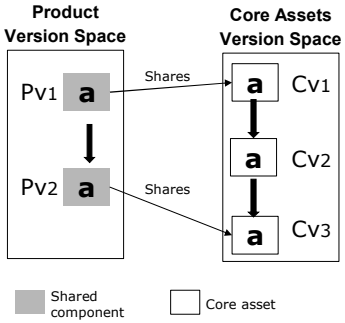
Reuse is a central theme of product line engineering, so any versioning system that would support product lines natively should support artifact sharing within the version system. Our approach uses *shared components* to enable reuse. Shared components provide an indirect reference to material stored elsewhere in the versioning system.

Modeling SPL & Sharing of Assets



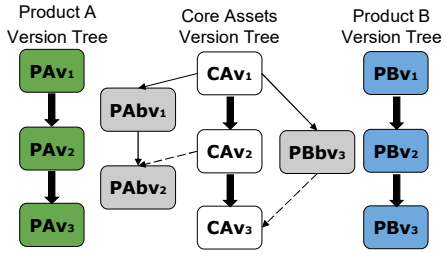
Evolution

Managing the evolution of a product line is challenging because of the independent evolution of products, core assets, and their interactions. We think of product line evolution as a two dimensional problem where one dimension is time (artifacts changing over time) and the other is space (artifacts varying between products).

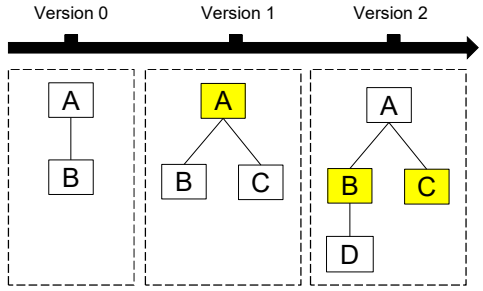


In the example to the left, the product is using shared component "**a**" from the core assets. However, as the product moves from version "**Pv1**" to version "**Pv2**", the version of the shared component changes from "**Cv1**" to "**Cv3**"

Below is a version tree of products and core assets. The larger black arrows indicate the main development trunk while the smaller black arrows indicate branches. The dotted lines indicate a merge.



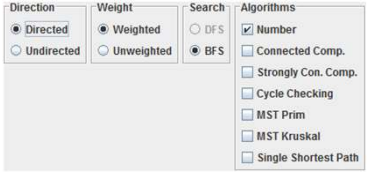
Versioning Model



Above depicts a Product Versioning Model. A version represents a state of the project tree and its artifacts. Each box represents a file or directory. A colored box indicates a file or directory that has changed since the previous version.

EVALUATION

We evaluate our prototype using the Graph Product Line (GPL) [3], 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 the SPLE research community. Our goals are to determine if our prototype can model a realistic software product line, capture its evolution, and evaluate change propagation between core assets and products.



Left is a GUI screenshot of a GPL program option screen which provides a means of specifying the implementation of a variety of GPL instances.

RESULTS

We evaluate our prototype with the GPL to confirm different change propagation cases. The prototype supports all the cases listed in the table below. Bold items are concrete components, while items in non-bold characters are shared components. The ' and * represent different changes.

	Before		After	
Case	Core	Product	Core	Product
1	**a′**	a	**a′**	a′
2	**a′**	a*	**a′**	a*′
3	**a′**	a*	**a′**	a′
4	**a**		**a**	a
5	**a**	a*	**a***	a*
6	**a′**	a*	**a′***	a*
7	**a′**	a*	**a***	a*
8		**a**	**a**	a

CONCLUSION

We have described an approach that is capable of versioning multiple types of product line projects including Graph Product Lines. It has a versioning 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.

BIBLIOGRAPHY

[1] P. Clements and L. M. Northrop. *Software Product Lines: Practices and Patterns.* Addison-Wesley, 2002.
[2] D. Weiss and R. C T. Lai, *Software Product Line Engineering.* Addison-Wesley, 1999
[3] Roberto E. Lopez-Herrejon and Don S. Batory. 2001. A Standard Problem for Evaluating Product-Line Methodologies. In Proceedings of the Third International Conference on Generative and Component-Based Software Engineering (GCSE '01), Jan Bosch (Ed.). Springer-Verlag, London, UK, 10-24.