# Ideas on How Product-Line Engineering Can be Extended*

Jeffrey M. Thompson and Mats P.E. Heimdahl
Department of Computer Science and Engineering
University of Minnesota
4-192 EE/CS; 200 Union Street S.E.
Minneapolis, MN 55455 USA
+1 (612) 625-1381
{thompson,heimdahl}@cs.umn.edu

## Abstract

*Product-line engineering can result in cost savings and increases in productivity. In addition, in safety-critical systems, the approach has the potential for reuse of analysis and testing results which can lead to a safer system. Nevertheless, there are times when it seems like a product family approach* should *work when, in fact, there are difficulties in properly defining the boundaries of the product family.*

*In this paper, we present a position on* n-dimensional *and* hierarchical *product families, which we have recently introduced. This paper focuses on our initial thoughts on how making n-dimensional and hierarchical families has the potential to affect the product-line development process as well as how using this approach might enable more organizations to use product-line approaches.*

## 1 Introduction

Although one of the main barriers to the use of product family techniques is one of process and organizational acceptance, technical issues have not been completely solved for product-line engineering. The techniques available work best for cohesive product families, where the variabilities do not have complex interdependencies. When this is not the case, it can be difficult to apply the product family approach even though there might be significant commonalities between the members of the family. As an alternative, we propose to view the families themselves in a multi-dimensional and hierarchical fashion. This helps us to deal with existing problems, for example, near commonalities, and helps to extend the approach to domains which, traditionally, would be difficult for product-line engineering.

*This work as been partially supported by NSF grants CCR-9624324 and CCR-9615088, and by NASA grants NAG-1-2242 and NCC-01-001.

We introduced the notion of *n-dimensional* and *hierarchical* product lines in [9]. In this paper, we present a position on how this techniques may affect the software product-line development process and how it may assist in the adoption of software product-line engineering within organizations.

## 2 Product-line engineering

The notion of a product family was introduced by David Parnas in [8]. The FAST (Family-oriented Abstraction, Specification and Translation) process advocates the creation of a specialized language for the domain for each family [10]. A similar process is mentioned by Campbell *et al.* in [3] and by Lam [6]; the differences between these works are primarily in the sort of artifacts produced by the domain engineering.

Current techniques for product-line engineering work well (1) the systems in the family share significant commonalities, and (2) The variabilities which define each family member have a straightforward decision model, i.e., it does not require many complicated rules to describe how the variability values are assigned to produce each family member. The first point describes the essential feature of product families that Parnas noticed in his work. However, the second point originates in the practical experience of many researchers who have labored to construct software product-lines. In general, the simpler the relationships among the variabilities, the easier it is to construct the product family.

**n-Dimensional product families:** Attempts have been made to organize the product family requirements in a hierarchical fashion [7, 8, 5, 6]. Lutz noted in her attempt to organize the variabilities into a tree that "there were several possible trees, with often no compelling reason to select one possible tree over another" [7].

Brownsword and Clements present a shipboard command and control systems family which contained 3000-5000 parameters of variation for each ship [2].

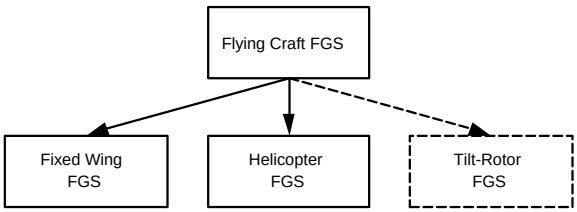


**Figure 1. FGS product family covering flying craft**

They state that "the multitude of configuration parameters raises an issue which may well warrant serious attention." In addition, they present three different views of the architectural layering of the base system in the case study which "do not conflict with each other; rather they provide complementary explanations of the same ideas."

Both these examples, as well as our own experience, illustrate the fact that often a product family is multi-dimensional; therefore, a hierarchical decomposition is not sufficient to capture the structure of the domain. We call such domains *n-dimensional product families.*

**Hierarchical product families:** Suppose that a company wished to construct a flight guidance system (FGS) for both fixed-wing aircraft and helicopters[1]. The FGS is responsible for issuing commands that keep the aircraft level, cause it to climb or descend, and so forth. Furthermore, the FGS must interact with other airborne systems. Many of the tasks that the system has to perform might be common across these two radically different aircraft. Nevertheless, the actual control of the aircraft is very different. Therefore, developing a single set of commonalities and variabilities which span this entire domain is difficult.

Some would argue that the FGS example is simply too diverse to be considered a product line. However, it is clear that these systems share much in common, which is the most important criterion for being a family. Thus, we propose the concept of a *hierarchical product family*.

Most previous attempts at product family structuring have focused on hierarchically grouping the *variabilities* while the *commonalities* remain the same for all family members [7, 6]. Notable exceptions are Parnas [8] and Brownsword and Clements who noted in their case study at CelciusTech [2] that sometimes product-lines exist within the main product line.

In our approach, *additional commonalities* which are *unrelated* to the parent product family can be added in the sub-families. The hierarchical decomposition of the FGS family is shown in Figure 1. Thus, the helicopter sub-family can have significantly different requirements than for fixed-wing aircraft, yet share many things in common as well.

[1]We would like to thank Steven P. Miller of Rockwell-Collins Inc. for this example

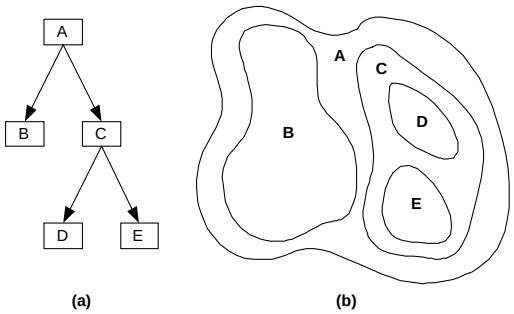


**Figure 2. Hierarchical decomposition and subset structure**

By structuring the requirements in this way, we focus on the structure of the domain itself. Furthermore, should the company wish to start building FGS systems for an entirely new set of aircraft, for example, tilt-rotor aircraft, this could be done while reusing many aspects of the FGS systems already implemented (Figure 1).

This will eventually effect the architecture and structure of the systems. For example, the product of the domain engineering for the parent family, Flying Craft FGS, might be a set of reusable components, whereas the product of domain engineering for the children might be a reference architecture or generation facility. The architectures for the fixed-wing aircraft and the helicopters could differ significantly and use the components from the parent family in different ways.

## 3 Structuring technique

One way to view a product family is as a set, where the boundaries of the set are determined by the commonalities, and the individual members of the set are distinguished by the values of their variabilities. Furthermore, the family may be undefined at some points within the boundaries due to, for example, illegal or non-functional combinations of variability values. This section shows how to use set theory to reason about the structure of product families (a more extended version can be found in [9]).

The most basic structure that can be represented with the set theoretic approach is the subset (Figure 2). This corresponds to a hierarchical decomposition of the family. The general requirements for any family **E** which is a subset of another family **C** is as follows:

- **E** must include all of the commonalities and variabilities in **C**; however, **E** may restrict the range or options available in the variabilities.
- **E** can add additional commonalities and variabilities which are not present in **C** as long as the additional they do not conflict with the commonalities or variabilities in **C**.

The first criterion is straightforward and necessary for the subset **E** to be completely contained within **C**.

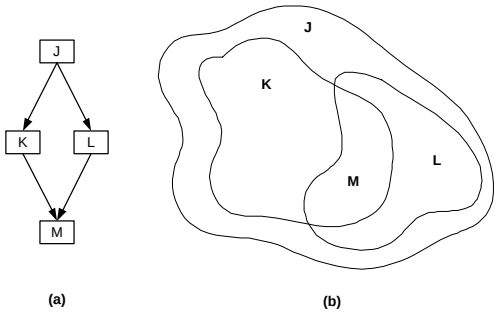


**Figure 3. Set intersection and non-hierarchical structure**

The second criterion defines the fact that **E** may wish to refine or restrict the values of the variabilities of **C**. It is possible for this refinement to result in an additional commonality. Additional commonalities can also be added which are unrelated to the parent family. For example, it is likely that the family of helicopters will need different commonalities than the family of fixed-wing aircraft. Finally, it is possible to add additional variabilities.

Another structure that can be represented using a set-theoretic approach is that of set intersection. The ability to represent a set intersection distinguishes this approach from the purely hierarchical structures which have been applied by others. This is shown in Figure 3. The constraints on any family **M** which is a subset of families **K** and **L** are as follows:

- **M** must include all the commonalities and variabilities of both **K** and **L**; however, it may restrict those variabilities as above for subsets.
- **M** may introduce additional commonalities and variabilities which are not present in **K** or **L**.

These structures can be used to reason about families which are both *n-dimensional* and *hierarchical*.

## 4 A brief evaluation

The structuring technique presented results in the creation of more families within the domain than with a traditional approach. However, these sub-families are more cohesive and simpler than would be the case if we created just one top level-family. We believe that this provides several benefits. First, the top-level family can now be much broader than was previously possible. Second, the overall family can be expanded and contracted by adding and subtracting sub-families. Finally, these techniques will allow a family to be more easily refactored as the definition of the family evolves over time.

The ability to draw a larger product family was an essential requirement for the structuring technique. This grows out of our own experiences with mobile robotics [4], where we had difficulty in applying the product family approach. This difficulty stems from the fact that the mobile robotics domain is both *n-dimensional* and *hierarchical*.

The mobile robotics domain breaks down along two clear dimensions: the hardware platform and the desired behavior. Each hardware platform conforms to a basic specification: it can move forward and backward, turn left and right, sense whether or not an object is in front of it. The hardware platform may also be equipped with a variety of sensors and actuators that give it additional capabilities; and, the various sensors differ greatly in the speed and accuracy with which they provide information. Thus, on the hardware side, there are many different configurations that must be modeled.

On the behavior side, we can imagine that a basic behavior might be a random exploration where the primary goal of the robot is collision avoidance and recovery. More complex behaviors can be added, for example, wall following, going through doors, and finding particular objects. Furthermore, those behaviors may be composed and combined to form a composite behavior. We might envision a behavior which includes the door navigation, a wall following behavior, and a high-level planner. The high-level planning behavior needs to communicate with the random exploration, door navigation, and wall following to direct the robot towards high-level goals. However, if the robot collides with an obstacle, then the lower level behavior will take over and recover from the collision. Thus structure of the behavioral dimension is much different from the hardware dimension and resembles Brooks' subsumptive architecture [1].

Certainly, a domain such as mobile robotics which absolutely requires n-dimensional and hierarchical product families will necessarily be more complex than a domain that does not require these techniques. Nevertheless, any domain can benefit from reuse of the artifacts at the top of the family hierarchy and a more traditional cost-benefit will exist towards the leaves of the family (along each particular dimension).

Another benefit of the technique is the ability to expand and contract the family as necessary. This ability is essential because it allows a more incremental development of product-lines than is facilitated by current approaches. Furthermore, it facilitates *family refactoring*; that is, the family can be redefined more easily as the product line evolves over time. Thus, this structuring technique has much potential to increase the usefulness of the product family approach.

One of the barriers to traditional product family approaches is that the whole organization must change to accommodate product-line oriented development. Many resources are required to develop the domain engineering support for the entire product line while at the same time continuing to produce products for existing customers. Our approach allows an organization to start out with a high-level product family and reuse just a few key pieces between the major product areas. As the payoff from this reuse makes more organizations resources available, the organization can then afford to make the family more rich (by refactoring and/or adding sub-families) and thus achieving more payoff from the effort.

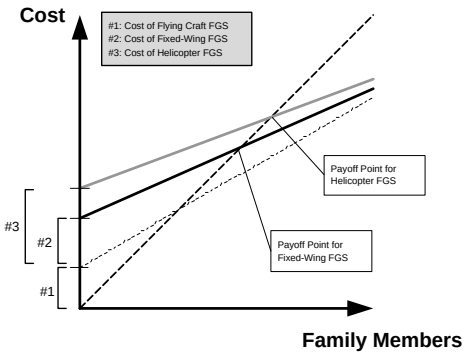


**Figure 4. Cost-benefit of the FGS Family**

Of course, these benefits do not come for free. The broader and more flexible view of product families allowed by our techniques will result in families which are more complex than traditional families. In addition, because of this broader view, it may be more difficult to determine what constitutes a viable family under our approach. Almost anything is related in some fashion or other and it may be difficult for organizations to decide when to define an encompassing family for a particular group of subfamilies. Nevertheless, we feel that these techniques hold promise and may serve to advance the frontiers of product-line engineering.

The cost-benefit analysis of our product-line engineering approach is more difficult because one must not only consider the cost of developing domain engineering support of the particular sub-family in which the member resides, but also all sub-families above that one in the product family hierarchy. For example, the cost-benefits for the FGS family is shown in Figure 4. The payoff for the fixed-wing FGS is shown by the thick black line and the payoff for the helicopter FGS family is shown by the thick gray line. As the figure demonstrates, the payoff point for the two subfamilies is different, because the cost of implementing each over and above the functionality provided by the flying craft FGS family is different. As the structure of the family becomes more complex, for example, through the creation of a deeper hierarchies and/or the use of multiple dimensions with constraints between them, this relationship will become more complex.

## 5 Conclusions and future work

In this paper, we have reflected on some current issues with product-line engineering and presented some ideas about our initial attempts towards extending the product family approach to better address these issues as well as support what we call *n-dimensional* and *hierarchical* product families.

When we examined problems that we and others have had in developing product families, we concluded that the difficulties in modeling near-commonalities and variability dependencies stemmed more from the structure of the domain itself and the lack of an ability to adequately capture that structure. Further, the simple structuring technique that we proposed in [9] is based on set theory, which provides a simple and elegant solution. This allows the analyst to capture the structure of the domain and not be biased by implementation or design concerns. This approach allows thinking about *n-dimensional* and *hierarchical* product families and encourages many different views of the system.

In the future, we intend to continue developing this approach to product-line engineering. We are already working on several industrial-sized case examples with good initial results. We look forward to reporting the outcome of that work, as well as presenting a more detailed view of this approach. Furthermore, in the future we intend to provide a more detailed description of the formal foundations of this approach and how it could be leveraged if the family members were specified in a formal specification language.

## Acknowledgements

The authors wish to thank Steven P. Miller at the Rockwell-Collins Advanced Technology Center for his thoughtful comments and encouragement with regards to this work as well as his insights on how the structuring techniques presented here might be applied to the systems manufactured by Rockwell International.

## References

[1] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, March 1986.

[2] Lisa Brownsword and Paul Clements. A case study in successful product line development. Technical Report CMU/SEI-96-TR-016, Software Engineering Institute, Carnegie-Mellon University, October 1996.

[3] G. Campbell, J O'Connor, C. Mansour, and J. Turner-Harris. Reuse in command and control systems. *IEEE Software*, 11(5):70–79, September 1994.

[4] Debra M. Erickson. Structuring formal requirements specifications for reuse: A mobile robotics case study. Masters Project, University of Minnesota, April 2000.

[5] Juha Kuusela and Juha Savolainen. Requirements engineering for product families. In *Proceedings of the Twenty-Second International Conference on Software Engineering (ICSE'00)*, pages 60–68, June 2000.

[6] W. Lam. Creating reusable architectures: Initial experience report. *ACM SIGSOFT Software Engineering Notes*, 22(4):39–43, 1997.

[7] Robyn R. Lutz. Extenting the product family approach to support safe reuse. *Journal of Systems and Software*, 53:207–217, 2000.

[8] D.L. Parnas. On the design and development of program families. *IEEE Transactions on Software Engineering*, 2(1):1–9, March 1976.

[9] Jeffrey M. Thompson and Mats P.E. Heimdahl. Extending the product family approach to support n-dimensional and hierarchical product lines. In *The Fifth IEEE International Symposium on Requirements Engineering*, August 2001.

[10] David M. Weiss and Chi Tau Robert Lai. *Software Product Line Engineering: A Family-Based Software Development Process*. Addison-Wesley, 1999.