Multistage Configuration Trees for Managing Product Family Trees

Charles W. Krueger
BigLever Software
10500 Laurel Hill Cove
Austin, TX 78730 USA
+1-512-426-2227
ckrueger@biglever.com

ABSTRACT

It is not unusual for commercial product line organizations to manufacture millions of product instances every year, in thousands of different "flavors". The scale and scope of diversity in product lines of this size can be high, creating significant challenges to engineers implementing the product line, product marketers defining the space of available products, and customers selecting from available products. Companies often organize their products into a *product family tree* to provide clarity about their product groupings and offerings, better enabling their customers to effectively navigate among the huge number of offerings and to efficiently converge on a suitable product instance. This paper describes a 2nd Generation Product Line Engineering (2GPLE) feature modeling structure called a *multistage configuration tree* that supports the engineering, deployment and maintenance of complex product family trees. Feature selections and downselections are incrementally staged throughout the nodes in a product family tree. Feature decisions made at any node are inherited by all descendants of that node, thereby defining a product family subtree.

Categories and Subject Descriptors

D.2.2 [**Design tools and techniques**]: product line engineering, software product lines, feature modeling, multistage configuration trees, product family trees.

General Terms

Design, Economics, Management, Measurement, Theory.

Keywords

Multistage Configuration, Staged Configuration, Product Line Engineering, Product Family Tree, Systems and Software Product Lines.

SPLC 2013, August 26 - 30 2013, Tokyo, Japan

ACM 978-1-4503-1968-3/13/08…$15.00.
http://dx.doi.org/10.1145/2491627.2491648

1. Introduction

In successful commercial product line organizations, the number and diversity of products deployed can grow to be extremely large, partially due to the efficiencies made available from Systems and Software Product Line Engineering (PLE) tools and methods. In market segments ranging from industrial pumps to automobiles, it is not unusual for companies to manufacture millions of product instances every year, in thousands of different "flavors".

The extreme scale and scope of diversity in product lines of this size creates significant challenges to engineers implementing the product line, product marketers defining the space of available products, and customers selecting from available products. To provide order and clarity about their product groupings and offerings within this type of complex product space, companies often organize their products into a *product family tree*, thereby enabling their customers to effectively navigate among the huge number of offerings and to efficiently converge on a suitable product instance.

For example, an automotive manufacturer might hierarchically structure their entire product line portfolio with millions of instances into a family tree with 5 levels:

- **Platforms**. A platform is a family of vehicles of similar size and structure that can be manufactured in the same assembly plant. Examples might be pickup trucks, large sedans, and small coupes.
- **Programs**. A program is subfamily of vehicles within a single platform, known by consumers as the "model" and often found in nameplate display on the back of the vehicle .
- **Regional programs**. A regional program is a subfamily of vehicles within a single program, manufactured for the legislative, geographic, climate, cultural, and marketing characteristics of a particular country.
- **Trim levels**. A trim level is a subfamily of vehicles within a single regional program, representing different tiers of capabilities, accessories, and associated cost. Trim levels are marketed using terms such as base, standard, and luxury.
- **Vehicle instances**. A vehicle instance is a subfamily member within a single regional program trim level. Characteristics of a vehicle instance are determined by the consumer-selectable options available on a particular trim level.

In contrast to the way that the PLE community traditionally focuses on techniques for selecting and solving for features on a particular product instance, product line organization with large family trees expend most of their effort determining which features will not be available within a subfamily. As an engineering leader within one such automotive manufacturing organization described it, "we are more focused on identifying the features that we don't want on a particular platform, program or trim package, than we are on specifying the features we want configured on any particular product instance."

To address this need for managing the feature modeling structure within a product family tree, we have designed, implemented and deployed into commercial practice a technology and methodology referred to as *multistage configuration trees* in the BigLever Software Gears 2GPLE tool and lifecycle framework[1][2]. Multistage configuration trees extend the PLE concepts and constructs of featuring modeling[3] and staged configuration[4] to support the engineering and deployment of systems and software product line engineering assets for product lines with complex product family trees[5].

In 2GPLE terminology, a *feature model* declares the full collection of feature choices available in a product line and a *feature profile* defines the fully bound collection of feature choices made for a particular product instance in the product line. In multistage configuration trees, the feature model serves as the root of the tree and fully bound feature profiles are found at the leaves of the tree. *Partially bound feature profiles*, where some feature decisions have been made or restricted and other feature choices remain available, are present either as internal or leaf nodes of the tree.

One of the central properties of multistage configuration trees is that any connected path from the root node to a leaf node must be a monotonically decreasing (i.e., monotonically not increasing) sequence in the space of available feature choices. That is, children must honor the feature decisions made by their ancestors and may optionally decide to make additional feature decisions that further constrain the space of available feature choices. The monotonically decreasing property of multistage configuration trees assures that each node in a tree defines a subfamily, where all descendants of a node inherit the feature decisions from that node and its ancestors.

The addition of multistage configuration trees into 2GPLE, in order to better manage the engineering and deployment of an organization's product family tree, introduces new challenges and opportunities which are discussed throughout the remainder of this paper. These include the use of partial profiles to perform partial configuration of assets, managing the evolution of multistage configuration trees, recursive application of multistage configuration trees in hierarchical product lines, and tool automation to support and enforce monotonically decreasing feature spaces.

2. Multistage Configuration Trees

Multistage configuration trees[1] comprises three kinds of models:

- *Feature models* are located only at the root of the tree. As with conventional 2GPLE, a feature model declares the full collection of feature choices available in a product line.
- *Feature profiles*, or *fully bound feature profiles*, are located only at leaves of the tree. Just like conventional 2GPLE, a *feature profile* defines the fully bound collection of feature choices made for a particular product instance in the product line. Feature profiles never have children in a multistage configuration tree since no further refinements or feature choices are possible.
- *Partial profiles*, or *partially bound feature profiles*, are found at internal nodes and leaves of the tree. Partial profiles are a special kind of profile, where you can bind some feature decisions (as in a conventional feature profile), partially restrict the available choices some feature, and explicitly leave other feature choices unbound.

Note that the root *feature model* in a multistage configuration tree is a degenerate case of a *partial profile* with no decisions made. A leaf *feature profile* is another degenerate case of a *partial profile* with all decisions made.

2.1. Partial Profiles

A partial profile is more than just a partially filled out feature profile. There are special modeling constructs and semantics associated with the feature decisions that remain unbound.

In a conventional fully bound profile, there are two states for each feature decision, *undefined* and *defined*. Undefined means that the modeler has not made a decision about a feature, while defined means that the choice for a feature is fully defined.

For partial profiles, there is a third state possible for any feature, *unbound*. Unbound means that the modeler has explicitly made their decision to leave this decision open, so that other descendants in the multistage configuration tree can make their own – possibly different in different subfamilies – decision about this feature choice.

For feature choices in an unbound state, it is also possible to restrict the space of candidate feature selections through *downselection*. Downselection reduces the available diversity for a feature, while still leaving some diversity open in the unbound decision.

The semantics of downselection on the feature types in a feature model will obviously depend on the feature modeling language. In this paper, we focus on the feature model language and semantics of the Gears PLE Lifecycle Framework from BigLever Software. Following are examples of downselection on the discrete feature types in Gears, *enumerations*, *sets*, and *booleans*.

[1] Patent pending

- Enumerations. An *enumeration* is a feature with a discrete, enumerated set of member choices. One and only one member is selected to define an enumeration feature choice.

  Downselection in an enumeration type feature means removing one or more of the members from consideration. A downselected member can never be chosen as the value for that enumeration.

- Sets. A *set* is a feature with a discrete, enumerated set of member choices. Zero or more members are selected to define a set feature choice.

  Downselection in a set type feature means removing one or more of the members from consideration. A downselected member can never be chosen as a member of the full bound value for that set.

  Inverse to downselection, it is also possible to partially select some of the set members prior to fully defining the value for that feature. A selected member must always be chosen as a member of the fully bound value for that set.

- Booleans. A *boolean* feature is simply a special case of an enumeration, with two members: true and false. Since downselection of one of its members would fully determine the value of a boolean, downselection is neither needed nor supported on boolean type features.

Figure 1 shows an example of a downselection for an enumeration feature type. As indicated visually by the red strikethrough, the *DualZone* member of the unbound *ZoneType* enumeration feature in a partial profile has been downselected and no longer available for inclusion in partial or fully bound profiles in descendants in a multistage configuration tree.

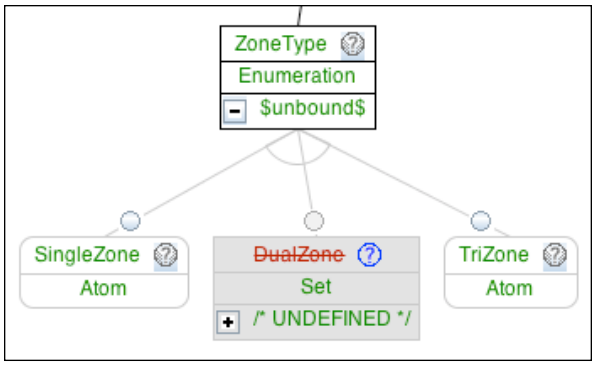


Figure 1. Unbound Enumeration with Downselection

Figure 2 shows an example of a downselection for a set feature type. As indicated visually by the red strikethrough, the *Attitude* member of the unbound *DualZone* set feature in a partial profile has been downselected and no longer available for inclusion in partial or fully bound profiles in descendants in a multistage configuration tree.

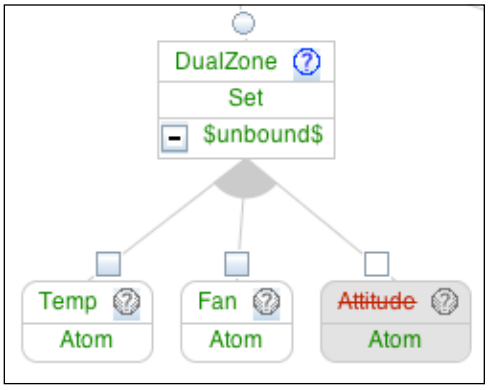


Figure 2. Unbound Set with Downselection

2.2. Inheritance Rules

The partial profiles and fully bound profiles in a multistage configuration tree are governed by inheritance rules. Feature selections and downselections made at any node in a multistage tree are inherited by all descendants of that node, thereby defining a product family subtree.

The feature decisions – selections, downselections and transitions from unbound to defined – along any connected path from the root node to a leaf node in a multistage configuration tree must be monotonically decreasing. Child profiles inherit and may not override the feature decisions made by their parents and ancestors on the path to the root. Child profiles may optionally decide to make additional feature decisions that further constrain and define the space of available feature choices.

Any profile that defines all remaining unbound feature choices becomes a leaf profile in its multistage binding tree. In theory, a fully defined profile could have a monotonically decreasing child that was identical, but this redundancy serves no purpose, so we disallow fully bound feature profiles from having children.

The monotonically decreasing property of multistage configuration trees assures that each node in a profile tree defines a subfamily of partially bound and fully bound profiles. Within a subfamily all descendant of a profile node inherit the feature decisions from that node and its ancestors. These inherited feature decisions determine the commonality of shared feature decisions within the subfamily.

Figure 3 shows a multistage configuration tree for managing a simple *HomeClimateControl* system in the Gears *production line* browser. The root of the tree – the feature model – is the second item in the list, labeled *Features*. There are two child partial profiles of the root, *Automatic* and *Manual*, that contain subfamilies for automated home climate control systems and for manually controlled home climate control systems. Within the *Automatic* family, there are three child members that are fully defined feature profiles: *AutoDualAllHeatCool*, *AutoTriZoneHeatCool* and *NewPrototype*. Similarly, the *Manual* family has two child members.

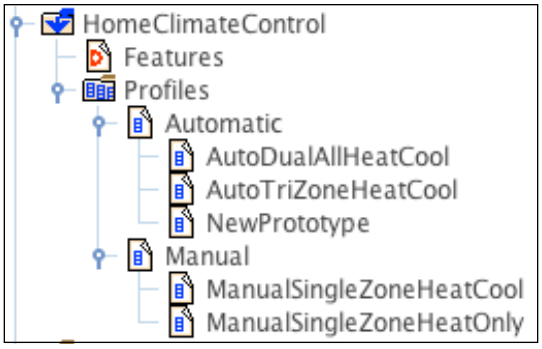


Figure 3. Multistage Configuration Tree for HomeClimateControl

Figures 4, 5, and 6 show a monotonically decreasing path in the multistage configuration tree in Figure 3, starting from the root *Feature* model in Figure 4, to the *Automatic* partial profile in Figure 5, to the leaf feature profile *AutoDualAllHeatCool* in Figure 6.

The feature decision of *Auto* for the *SystemType* enumeration in the partial profile in Figure 5 is inherited by the child profile in Figure 6, where the *SystemType* feature and its decision radio button are grayed out to indicate that the decision cannot be changed in the child profile due to the inheritance rules. The *Automatic* subfamily tree defined by the partial profile in Figure 5, leaves the decisions about *ZoneType* and *HVACType* fully unbound, so the full combinatoric space of possibilities from those two features is available in the subfamily.

The *AutoDualAllHeatCool* profile shown in Figure 6 fully binds all of the feature decisions left unbound in its parent. Therefore, this profile becomes a fully bound leaf node in the tree.

3. Product Line Scoping Revisited

Conventional wisdom from the product line engineering literature makes a good argument that when the available feature diversity within a product line is high and the commonality is low, the benefits of leveraging the small amount of commonality during the engineering process may be negated by the overhead of managing the variability. In these cases, the conventional guidance is to split the product line into multiple, smaller and more internally cohesive product lines that each possess higher ratios of commonality to variability[6].

The drawback of this approach is that after a product line is split into smaller and more cohesive families, these subfamilies become silos that can no longer take disciplined advantage of any commonality that exists among them. It suffers from the classic clone-and-own problem, but in this case for entire product families.

Multistage configuration trees offer an alternative to splitting large and diverse product families into multiple internally cohesive but isolated subfamilies. By keeping the multiple families organized in a multistage configuration tree, cohesive subfamilies can be grouped into subtrees within the same tree.

The multistage subtrees provide the cohesiveness and lower diversity of the smaller subfamilies through appropriate selections, downselections and unbound states in their parents and ancestors in the multistage configuration tree. The commonality among the diverse subfamilies, no matter how large or small, is shared from the common ancestor nodes on the path to the root, thereby avoiding the need to split into independent subfamilies and become divergent through clone-and-own. The benefits become clearly evident in very large product families, where decisions or changes made at higher levels in a multistage configuration tree are inherited by hundreds, thousands, or even millions of subfamilies and product instances.

4. Multistage Configuration Trees for Hierarchical Product Line Families

A core characteristic of 2GPLE is hierarchical product lines – the capability to hierarchically compose larger product lines from a collection of smaller product lines[3]. This is analogous to building a *system-of-systems* in one-of-a-kind systems engineering, but in this case each system and subsystem is a product line. The result is a *product-line-of-product-lines*. Multistage configuration trees can also be applied to managing the configuration of these hierarchical product line families.

The application of multistage configuration trees that we've discussed thus far in this paper have been applied within the context of a single product line, to structure subfamilies and family members within the product line. Hierarchical product lines introduce another mechanism to define a larger granularity product family, but in this case the family is defined in terms of variant assemblies of the different "flavors" offered by each of the composite product lines in the product line hierarchy. In commercial PLE practice, we are finding that multistage configuration trees applied to hierarchical product line families is essential in managing very large product lines of this form, such a automobiles which are comprised of approximately one thousand hierarchical subsystems.

At this point, we now have put four hierarchies into play.

- feature model trees for a product line
- multistage configuration trees applied to feature profiles for a product line
- product-line-of-product-line trees
- multistage configuration trees applied to hierarchical product line assemblies.

Fortunately, adding the last item to the list and as another hierarchy to the product line methodology is simpler than it might seem. The following example illustrates how multistage configuration trees help significantly with hierarchical product line families.

Starting with our previous example of the *Home Climate Control* system, we add two more product lines, *Home Security* with different capabilities for intruder detection and alerts, plus *Home Fire Protection* with different capabilities for fire detection, alerts and suppression. Each of these three product lines is offered in different configurations of features, similar to the product offerings shown for our home climate control system shown in Figure 3.

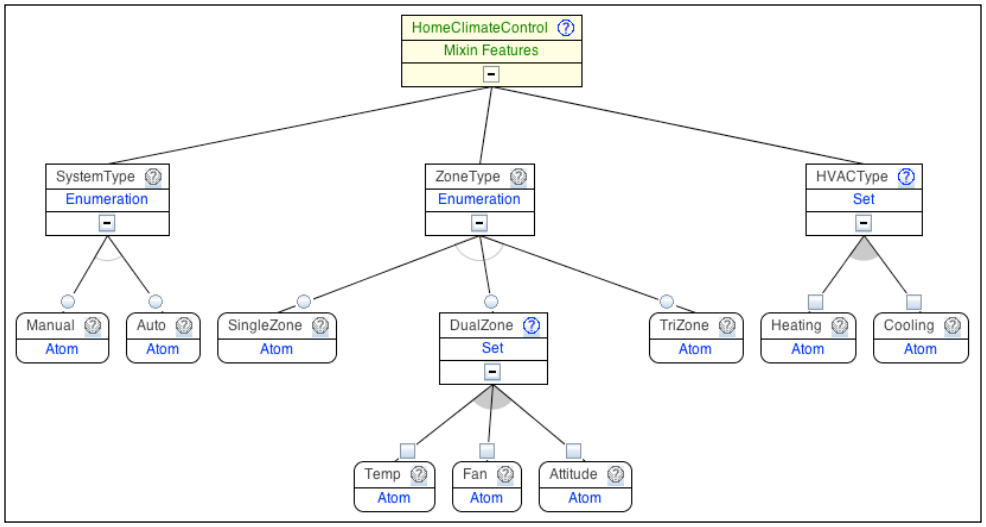


Figure 4. Feature Model

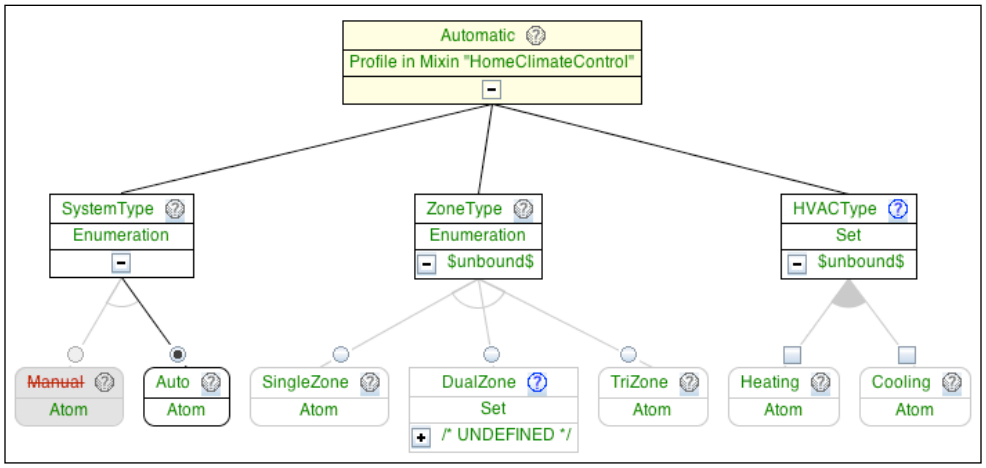


Figure 5. Automatic Partial Profile

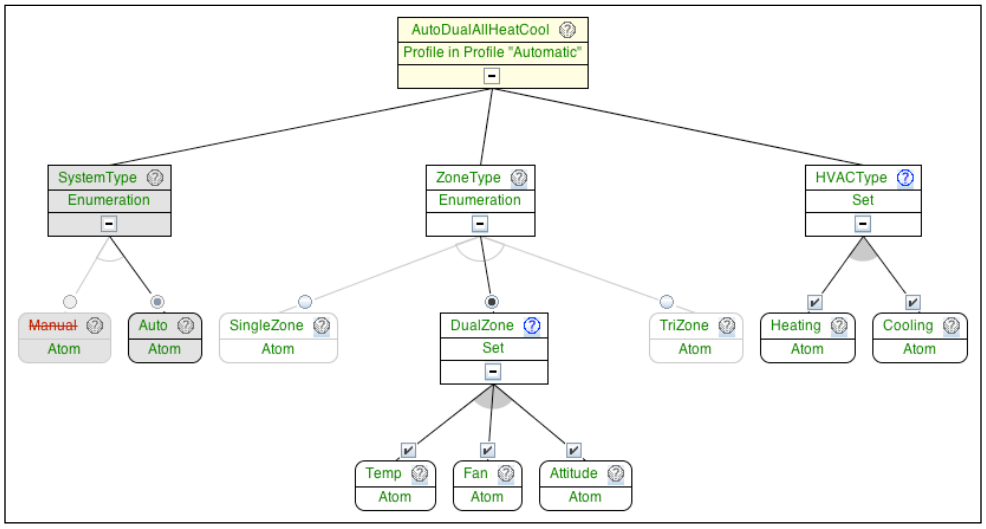


Figure 6. AutoDualAllHeatCool Fully Defined Feature Profile

To create the product line hierarchy for this example, these three product lines are composed into a larger product line called *Home Automation*. Figure 7 illustrate this product line hierarchy in Gears. *HomeAutomation* is the root product line and *HomeClimateControl*, *HomeFireProduction*, and *HomeSecurity* are *nested product lines*, indicating their composition in the higher level *HomeAutomation* product line.

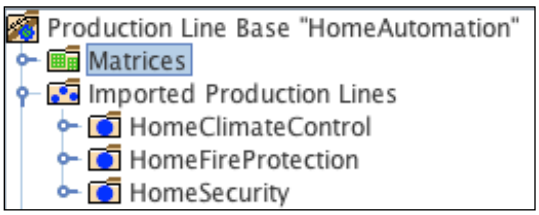


Figure 7. HomeAutomation Hierarchical Production Line

A product marketing role can now define the *Home Automation* products that will be offered to the market by enumerating them in a product Matrix. As illustrated in the matrix in Figure 8, the products are branded under the name of *ComfortHome,* with model numbers such CH10 and CH55.

Each product offering at the *Home Automation* level is defined as a named product row in the matrix. The columns correspond to the nested product lines and the values selected for each cell in a row are selected from the products offered in each of the nested product lines. For example, in the *HomeClimateControl* column you will recognize the product offerings we defined in Figure 3 and Figures 4 through 6.

The drawback of the management of the product line hierarchy in this monolithic listing is the same as the drawback discussed in the first part of this paper for managing a single product line as a monolithic collection of feature profiles. Particularly as the product line gets very large and the list of products in the product matrix gets very long, it becomes difficult to express and decipher the families and subfamilies within the space.

For example, in Figure 8, the *ComfortHome* product line is divided into 3 cohesive subfamilies for marketing purposes, but the only hint of these subfamilies is a weak naming convention in the left column: 10's, 20's and 50's. From the perspective of product marketing, of engineering, and of the consumer, there is no clear way to see or take advantage of the commonalities and better manage the variabilities within this product space.

Applying multistage configuration trees to the definition of hierarchical product lines addresses this deficiency.

Figure 9 shows a multistage configuration tree of matrices in Gears, analogous to the multistage configuration tree of feature profiles shown in Figure 3. Each of the named "grid" icons corresponds to a single-row matrix, similar to Figure 8.

The root matrix, *ComfortHome*, will have no decisions bound, analogous to a feature model. The three child matrices below the root, *Base*, *Mid*, and *Deluxe* group subfamilies and define selections, downselections in their respective matrices that are inherited by all descendants in the subfamily. The fully bound models, such as *CH10* and *CH55*, are at the leaves of the tree and contain fully bound matrix rows.

Note the small 'T' on some of the matrix icons. This indicates a *partially matrix*, meaning that the matrix is a *template* that contains unbound decisions. Fully bound matrices with all decisions made are shown as grid icons without a 'T'. Following the same inheritance rules as in multistage configuration trees for feature profiles, partial matrices will be be found at internal nodes or leaves in the tree, while fully bound matrices will be found only as leaves in the tree, with no children.

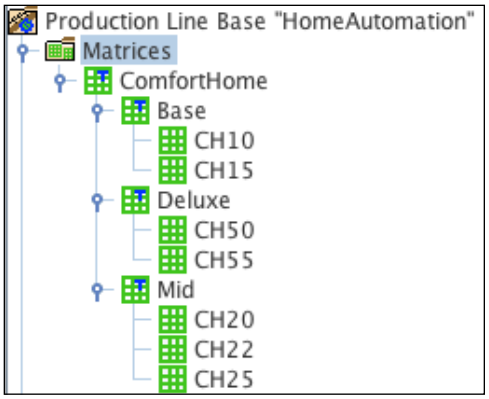


Figure 9. Multistage Configuration Tree for the HomeAutomation Family

ComfortHomeModels	HomeClimateControl	HomeFireProtection	HomeSecurity
CH10	Manual.Product/ManualSingleZoneHeatCool	$omitted$	HomeSecurity/OutsideAlarmOnly
CH15	Manual.Product/ManualSingleZoneHeatCool	HomeFireProtection/AudibleAlarmOnly	HomeSecurity/OutsideAlarmOnly
CH20	Automatic.Product/AutoSingleZoneHeatCool	HomeFireProtection/AudibleAlarmOnly	HomeSecurity/OutsideAlarmOnly
CH22	Automatic.Product/AutoDualAllHeatCool	HomeFireProtection/AudibleAlarmOnly	HomeSecurity/AlarmAndPoliceAlert
CH25	Automatic.Product/AutoDualAllHeatCool	HomeFireProtection/EmergencyResponderAlert	HomeSecurity/AlarmAndPoliceAlert
CH50	Automatic.Product/AutoTriZoneHeatCool	HomeFireProtection/EmergencyResponderAlert	HomeSecurity/AlarmAndPoliceAlert
CH55	Automatic.Product/AutoTriZoneHeatCool	HomeFireProtection/AlarmAlertSprinklers	HomeSecurity/AlarmAndPoliceAlert

Figure 8. Home Automation Product Definitions in a Matrix of Nested Product Lines

Figures 10, 12 and 15 show a monotonically decreasing path in the multistage configuration tree from Figure 9. Figures 11, 13 and 14 show intermediate states using Gears to create and edit this multistage configuration tree.

Starting with the *ComfortHome* root matrix in Figure 10, all of the decisions remain *unbound* as seen in the matrix cells for each of the nested product line columns.

Moving to Figure 11, the *Mid* partially bound matrix, the pulldown menu shows the available offerings from the nested *HomeClimateControl* product line. The selection being made is the Automatic subfamily of *HomeClimateControl*, which will limit future selection in the descendants of the *Mid* subfamily to only come from the *Automatic* subfamily of *HomeClimateControl*. This illustrates an interplay between the multistage configuration tree of the overarching *HomeAutomation* product line and its subordinate *HomeClimateControl* product line. Figure 12 shows the end result after making the *Automatic* selection.

Downselections in a partially bound matrix eliminate certain choices that are being offered from the nested product lines. Figure 13 shows the downselections that have been set on the *Mid* partially bound matrix. The intent of the product marketing role who set these downselections is to express that the midrange products in the product subfamily are constrained to never select from these downselected offerings. For example, in the center downselection dialog for *HomeFireProtection* in Figure 13, two choices have been eliminated for all descendants in the *Mid* subfamily. Removing the *$omitted$* option means that every *Mid* subfamily member must select one of the available *HomeFireProduction* options. That is, every *Mid* product member must provide some form of home fire protection.

Moving to the multistage configuration tree leaf *CH22*, a fully bound matrix definition, decisions inherited from the *Mid* parent can be seen in Figure 14. Because the *Automatic* subfamily from *HomeClimateControl* was selected in Figure 11, none of the *Manual* options are available. Furthermore, the *$omitted$* and the *AutoTriZoneHeatCool* choices were downselected in Figure 13, so they are not available in the Figure 14 selectable options for the *CH22* product.

Figure 15 shows the final result of the fully bound *CH22* leaf in the multistage configuration tree.

5. Evolution of Multistage Configuration Trees

Just like feature models, feature profiles, product profiles, PLE assets, variation points and other constructs in an operational product line, multistage configuration trees are subject to constant evolution to support the ongoing evolution of a product line. Whenever changes are made on the multistage configuration tree constructs in feature models, feature profiles, or matrices, the implications must be considered and propagated across the full multistage configuration tree. As a reminder, the new constructs that have been introduced are:

- unbound states on feature choices and matrix choices
- partial selections in partially bound profiles and matrices
- downselections in partially bound profiles and matrices

Fortunately the inheritance rules in multistage configuration trees provide clear semantics and guidance, as well as opportunity for automated impact analysis and support for interactive refactoring on widespread changes.

The evolutionary changes that need to be supported fall into two categories: reducing the space of variability in a multistage configuration subfamily by constraining available choices and expanding the space of variability in a multistage configuration subfamily by relaxing available choices.

5.1. Evolution when reducing the space of variability in a multistage configuration tree

The changes that will reduce the space of variability in a multistage configuration tree are:

- changing the state of feature or matrix choice from unbound to a partially or fully bound selection
- changing a feature or matrix choice from partially bound to fully bound
- applying additional selections in a partially bound feature or matrix choice
- applying a downselection to a feature or matrix choice

These changes that reduce the space of variability must conform to the inheritance rules and selections of the ancestors. If the change is made within an internal node in a multistage configuration tree, these tighter constraints need to be propagated down to the descendants in the subfamily. Because the change is narrowing the space of possible variability, some new decisions can be automatically determined and some existing decisions within the subfamily may become invalid, Therefore, automated propagation, semantic checking, and reporting is crucial.

At the time of this writing, Gears' propagation of reduced variability is fully automatic for cases where the original value of the ancestor and descendant are identical. In cases where the ancestor and descendant values are different, the propagation operation will leave the descendant's value unchanged and rely on semantic checks to report those descendants that no longer satisfy the monotonic decreasing property.

An upcoming release of Gears is planned to support an interactive option, where the user can incrementally resolve each violation detected during the top-down propagation operation. Any changes made by the user to a node in the multistage configuration tree will recursively invoke the propagation from that updated node, rather than to continue propagating the value from the original ancestor.

5.2. Evolution when expanding the space of variability in a multistage configuration tree

The changes that will expand the space of variability in a multistage configuration tree are:

- changing the state of feature or matrix choice from a fully bound to a partially bound or unbound selection
- changing the state of a feature or matrix choice from a partially bound selection to unbound
- reducing selections in a partially bound feature or matrix choice
- removing a downselection on a feature or matrix choice

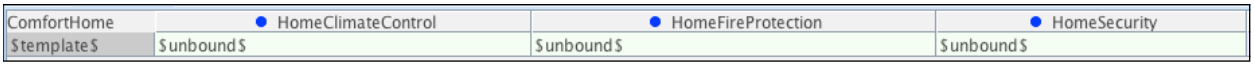


Figure 10. Root of the ComfortHome Multistage Configuration Tree

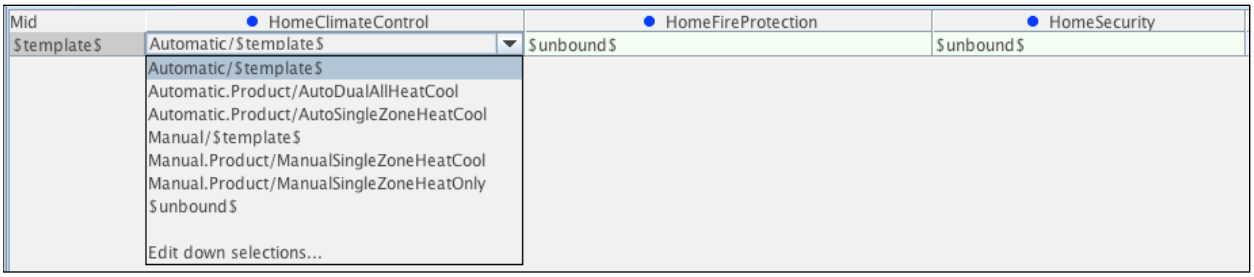


Figure 11. Selecting the Automatic HomeClimateControl Subfamily for the Mid Partially Bound Matrix

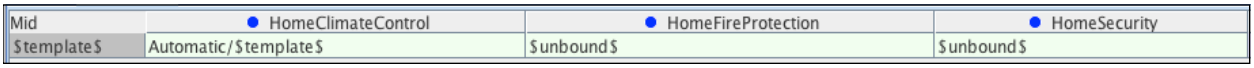
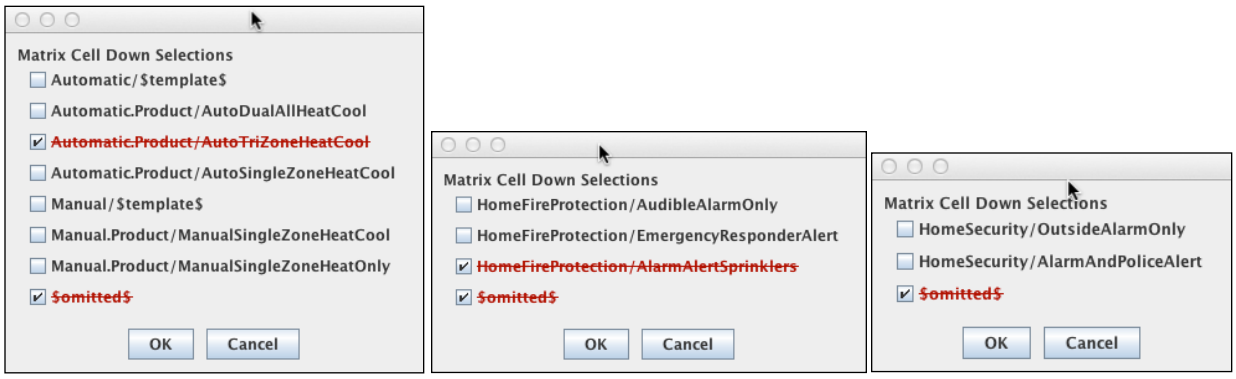


Figure 12. The Mid Partially Bound Matrix



HomeClimateControl HomeFireProtection HomeSecurity

Figure 13. The Mid Downselections

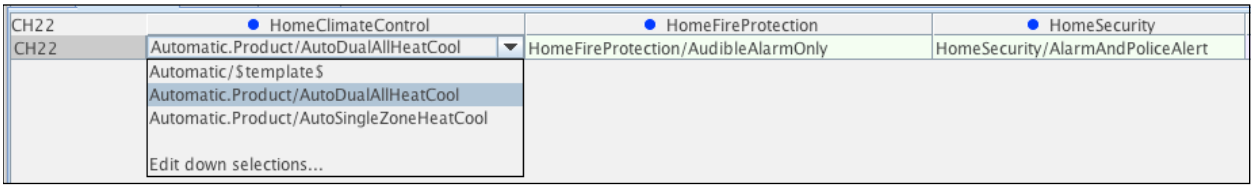


Figure 14. Selecting the AutoDualAllHeatCool HomeClimateControl for the CH22 Fully Bound Matrix

Figure 15. The CH22 Fully Bound Matrix

These changes that expand the space of variability must conform to the inheritance rules and selections of the ancestors. If the change is made within an internal node in a multistage configuration tree, the modeler might want the weaker constraints to be propagated down to the descendants in the subfamily. However, because the change is relaxing the space of possible variability, propagation is optional since none of the decisions within the subfamily will become invalid. Automated propagation would require human or heuristic guidance on how to deterministically relax existing choices within the subtree.

At the time of this writing, this support has not yet been implemented in Gears' multistage configuration trees.

6. Partial Configuration of PLE Assets

Supporting partially bound feature profiles and partially bound matrices as semantically valid constructs in a multistage configuration tree opens the possibility of doing partial configuration of PLE assets. Automated product configuration in 2GPLE is enabled by *variation points* in the PLE assets that define the mapping from feature selections in feature profiles to the configuration of some encapsulated feature-based diversity in the the asset[3].

There are three conditions to consider for a variation point when performing partial configuration based on partial profiles.

- If all feature values referenced by the variation point mapping logic are fully defined, then the variation point can be fully configured.
- If none of the feature values referenced by the variation point are defined, then the variation point remains unchanged in the configured asset.
- If some but not all of the feature values referenced by the variation point are fully defined, reduction on the mapping logic is required.
    - In some cases the reduction will fully resolve the mapping, in which case the variation point configuration can be completed.
    - In some cases, the reduction will leave the mapping unchanged, in which case the variation remains unchanged in the configured asset.
    - In some cases the mapping can be reduced to a simpler form, but the mapping cannot be fully resolved, in which case the variation remains in the configured asset with the simplified form of the mapping logic.

Fully bound matrices and profiles at the leaves of a multistage configuration tree are used to automatically configure assets for product instances. The need and the value of this is clear. Partially bound matrices and profiles in a multistage configuration tree can be used to automatically configure partially bound assets for product subfamilies. What is the meaning, need and value of this?

The answer will vary based on the asset type and how an organization uses the asset types in their engineering and business processes.

For example, partial configuration of source code that doesn't compile may not be of much interest. However, inspecting a partially configured set of requirements for a subfamily of low-end products and comparing that to a subfamily of high-end products could offer valuable insights about the common and variant properties of those two subfamilies. Unexpected content in the requirements for a subfamily might lead to a refinement in the definition of the product family feature profiles or matrix profiles, or it might indicate a defective mapping for a variation point that needs to be fixed.

7. Related Work

Staged configuration was introduced by Czarnecki, Helsen, and Eisenecker as a means of distributing the process for specify a feature model configuration among multiple roles across an extended timeline and workflow[4]. This concept has been formally modeled and enhanced in subsequent studies, including [7, 8, 9].

The focus of these studies is on how to support the sequential configuration of a feature model as it is handed off from one role to another during different stages of a business or engineering workflow. Multistage binding trees similarly enable incremental staging of feature selections along a branch in a multistage configuration tree, but they are intended for the organization of large, multi-level product line subfamilies, where staging across multiple roles or across workflows with extended timelines does not apply. Multi-level subfamilies within a multistage configuration tree are often engineered by a single individual or small team. Multistage configuration trees are intended to support concurrent development by different roles on parallel branches rather than sequential development by different roles along a single branch.

Reiser introduced the notion of product *sublines* in his PhD thesis[10]. His definition of a subline is based on product line scope subset, which is the equivalent to the definition of subfamilies in multistage configuration trees as being monotonic decreasing in variability relative to their ancestors. Reiser's work on sublines focuses on the hierarchical structure and staged configuration of assets rather than feature models or product profile models. Multistage configuration tree methods, on the other hand, focuses on the feature models and feature-based product models for a product line. With multistage configuration trees, the approach for deriving assets associated with any node in a multistage configuration tree is to apply 2GPLE automated configuration to the source repository of fully non-configured PLE asset supersets, using any partially bound or fully bound profile from a multistage configuration tree, as described in Section 6.

Elsner explored how staged configuration could be applied to the derivation of assets in a heterogeneous composition of multiple product lines (what 2GPLE refers to as hierarchical product lines)[11]. His work adheres to the seminal definition of staged configuration as a sequential and incremental configuration, distributed over roles and time[4]. Although this work does accommodate the need for the composition of a product-lines-of-product lines, we have shown the richer need and solutions for distinct support of multistage configuration trees within both a single product line and across the composition structures in a product-line-of-product-lines composition.

8. Future Work

As discussed in Section 5, Evolution of Multistage Configuration Trees, the inheritance rules in multistage configuration trees allow changes made close to the root of a multistage configuration tree to be propagated throughout all of the descendant subfamilies and family members. The benefit is that a single change to an ancestor node might be easily propagated to hundreds, thousands or millions of subtrees and instances. Of course, as would be affirmed by anyone who has worked to deliver products in a product line organization, this is also an opportunity for errors and unintended consequences.

Tools and techniques need to be identified that will provided detailed impact analysis and to also support human-guided propagation of changes. An example of the latter include marking nodes that require propagation, but allowing the owners of these subfamilies to respond lazily when they have the time, the resources and the need to do so.

Another alternative would be in the style of code refactoring tools, where the intent of a change could be declared before it was made, so that the tool could both perform the intended change (such as removing a downselection with the intent to also relax all compatible descendants), and then also guide the engineer with specific rationale in the refactoring at each of the candidate update sites during the propagation throughout the descendant subfamilies and family members.

The use of multistage configuration trees in commercial practice is relatively new, though the early experience with large scale product line organizations has strongly positive and is creating enthusiastic early adopters. The demand and opportunities for the industry to benefit from multistage configuration trees appear to be high as we introduce this new concept across a broad range of industry analysts, trade publications editors, and our existing and new customer organizations.

We anticipate that over time patterns, styles and scenarios of use will emerge. An early example of a common usage scenario that exposed a conceptual and tooling need was in the pattern of a user traversing up and down a multistage configuration tree, attempting to visualize and comprehend the monotonically decreasing selections and downselections for an *individual feature* along the path from the root of the family or subfamily tree to a descendant subfamily or individual product. We plan to capture pattens like this and reflect them back into the Gears tool, as well as to the PLE community as lessons learned and as best (and worst) practices.

References

[1] BigLever Software, “BigLever Software Gears,” http://www.biglever.com/solution/product.html

[2] Krueger, C. and Clements, P. “Systems and Software Product Line Engineering,” Encyclopedia of Software Engineering, Philip A. LaPlante ed., Taylor and Francis, 2013, in publication.

[3] Kang, K.; Cohen, S.; Hess, J.; Novak, W.; & Peterson, A. “Feature-Oriented Domain Analysis (FODA) Feasibility Study” (CMU/SEI-90-TR-021, ADA235785). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1990.

[4] Czarnecki, K., Helsen, S., Eisenecker, U. “Staged Configuration Using Feature Models”, Proceedings of the 2004 Software Product Line Conference (SPLC), Boston, MA, USA, August 2004.

[5] Flores, R., Krueger, C., Clements, P. “Mega-Scale Product Line Engineering at General Motors,” Proceedings of the 2012 Software Product Line Conference (SPLC), Salvador Brazil, August 2012.

[6] Clements, P., Northrop, L. Software Product Lines: Practices and Patterns, Sec 5.5, Addison-Wesley, 2002.

[7] Hubaux, A., Classen, A., Heymans, P. “Formal modelling of feature configuration workflows”, Proceedings of the 13th International Software Product Line Conference (SPLC), pages 221-230, San Francisco, CA, USA, August 2009.

[8] Bagheri, E., Di Noia, T., Gasevic, D., Ragone, A. “Formalizing interactive staged feature model configuration,” Journal of Software: Evolution and Process, Volume 24, Issue 4, pages 375–400, John Wiley & Sons, 2012.

[9] Schroeter, J., Lochau, M., Winkelmann, T. “Multi-perspectives on Feature Models”, Proceedings of the 15th International Conference on Model Driven Engineering Languages and Systems, pages 252-268, Innsbruck/AUSTRIA, September 2012.

[10] Reiser, Mark-Oliver, “Managing Complex Variability in Automotive Software Product Lines with Subscoping and Configuration Links”, PhD thesis, Technische Universität Berlin, December 2008.

[11] Elsner, Christoph, “Automating Staged Product Derivation for Heterogeneous Multi–Product-Lines”, PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2012.