Chapter 7
Visualizing Software Variability

Steffen Thiel, Ciarán Cawley, and Goetz Botterweck

What you will learn in this chapter
- *Core techniques in Information Visualization*
- *Using Visualization to support Software Variability*
- *Commercial and Prototype tools that utilize Visualization*

1 Introduction

Many of the expected benefits of software product line (SPL) engineering rely on an assumption that the additional up-front effort in domain engineering that establishes the product line produces a long-term benefit. The expectation is that deriving products from a product line during application engineering is more efficient than traditional single system development. However, to benefit from these productivity gains, it must be ensured that application engineering processes are performed as efficiently as possible. This has proven to be extremely challenging with industrial-sized product lines containing thousands of variation points, each of which can be involved in numerous dependent relationships with various other parts of the product line (e.g., [1, 2]). One method of addressing this challenge involves

S. Thiel (✉)
Furtwangen University of Applied Sciences, Furtwangen, Germany
e-mail: steffen.thiel@hs-furtwangen.de

C. Cawley
Dublin Institute of Technology, Dublin, Ireland
e-mail: ciaran.cawley@dit.ie

G. Botterweck
Lero-The Irish Software Engineering Research Centre, University of Limerick, Limerick, Ireland
e-mail: goetz.botterweck@lero.ie

supporting the SPL engineering activities by providing interactive tools that use, as a central principle, visualization techniques appropriate for the comprehension of large data sets and interrelationships.

Adopting visualization techniques in software product line engineering can aid stakeholders by supporting essential work tasks and enhancing understandings of large and complex product lines. This chapter presents visualization concepts, approaches, and implementations that are used to manage the application engineering phase of the SPL process.

2 Concepts and Techniques

2.1 Visualization

There has been extensive research into information visualization and its applications. Visualization has proven useful in enhancing cognition in numerous ways and application domains [3, 4]. This is particularly the case in relation to externalizing information, thus increasing the "memory" and "processing capacity" available to users, also by supporting the search for information and by encoding the information in a manipulable medium.

Visualization takes abstract data and gives it a form suitable for visual presentation. Such data can, for example, be explicitly collected from software or it can be codified by software engineers utilizing their own implicit knowledge. In this case, we often speak of software visualization, which can be seen as a subdiscipline of information visualization [5]. With suitable techniques, such software visualizations can also amplify cognition about large and complex data sets created and used in industrial software product line engineering.

2.2 Visual Reference Model

Figure 7.1 shows a visual reference model introduced by Card et al. [3]. This model provides a conceptual basis for many visualization approaches. *Source data* is transformed into a format (*data tables*) from which visual abstractions can be created. Various *views* can then operate over those abstractions, which provide the user with a rich interface. By allowing user interaction with the *view*, the different transformation steps can be altered in order to optimize the visualization for specific user tasks. This concept of interactive visualization forms the basis of many dynamic techniques aimed at providing cognitive support to stakeholders.

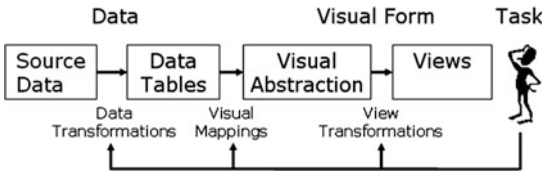


Fig. 7.1 Visual reference model [3]

Visualization techniques developed and discussed within the visualization community (e.g., [3–5]) can be leveraged to support variability management (e.g., [6–9]). By using such techniques, the expertise and experience of that community can be brought to bear on the complexity challenges that exist in that domain. Whereas most established variability management tools do not explicitly aim to utilize such techniques and expertise, recent research tools in that area are attempting to apply visualization concepts to their user interfaces.

2.3 Visualization Techniques

Fundamental visualization techniques and strategies that aim to support user cognition when dealing with large and complex data sets include *Focus+Context*, *Details on Demand*, *Degree of Interest*, *Color Encoding and Iconography*.

- *Focus+Context* describes the general ability to work at a focused level while maintaining the overall context within which you work. A number of techniques can be employed toward this goal such as *fisheye* (magnifying a specific area of a much larger display), *overview/outline windows* (providing a contextual understanding of a given display), and *distortion* (e.g., transparency). An extensive overview of these techniques is, for example, given in [3].
- *Details on Demand* refers to the facility whereby the stakeholder can choose to display additional detailed information at a point where this data would be useful. This point is decided by the user of the system. For instance, the ability to expand/collapse branches within a tree display, *incremental browsing* of such a tree and *filtering*, provides details on demand.
- *Degree of Interest* techniques highlight or expand relevant data with respect to the user's current point of interest. In particular, the degree of interest (as applied to certain parts of the data) can change while the user is navigating the data.
- *Color Encoding* and *Iconography* both serve to encode information visually and are used in conjunction with other techniques to provide additional data that can be identified through visual queries—identifying a visual pattern that will be used by a mental search strategy over a graphical visualization [3]. Examples include a green tick, red X, or a familiar icon.

3 Visualization Support for Software Variability

3.1 Representing Variability

In terms of how to represent and model variability, many SPL research approaches for variability management and product configuration focus on *features*, often represented by dedicated *feature models* (e.g., [10, 11]). *Feature models* usually describe available configuration options of an SPL in terms of "prominent or distinctive user-visible aspects, qualities, or characteristics" [11].

While viewing a product line as a collection of *features* has many advantages, there are some problems as well. Some of the problems include the difficulty in describing cross-cutting features and non-functional requirements, as well as the problems that arise in linking a feature to a concrete component (or set of components) that implement that feature.

3.2 Challenges and Approaches

There are numerous tasks to be performed by various stakeholders during the SPL engineering processes of domain and application engineering (cf. Sect. 3.3). Platform managers, domain engineers, product managers, application engineers, developers, and even customers all take on different roles in the process and require methodology and tool support that facilitates their specific tasks. In many of these cases, a feature model alone is either too detailed or not detailed enough. Using separate models allows different facets of the product line to be managed in a focused manner and supports stakeholder and task-specific representation and manipulation.

One approach to separating the different concerns of a software product line while providing relationships between various elements could be to describe the product line not only in terms of *features* but extend this description by taking *decisions* and *components* into consideration. A *decision model* would then capture a small number of high-level questions and provides an abstract, simplifying map onto *features*. The implementation of features by software or hardware components is then described by a *component model*.

Please note we use the concept of a *decision model* in the sense of a high-level *feature model* that sets the major context of the configuration by answering major questions such as if a particular product is "entry level" or if the product is planned to be introduced in a specific market (e.g., US, Japanese, or European market). In this sense, the *decision model* could be seen as containing the most important questions someone has to ask before configuring the more detailed and fine-grained *feature model*. This is different from other definitions of the term "decision model" in the product line literature, for example, the definition provided in [6].

These three models—the *decision*, *feature*, and *component* (DFC) *model*—can be used as a foundation to support variability visualization and product configuration. One characteristic of the DFC model is that the three underlying models are interrelated. For instance, making a *decision* might cause several *features* to become selected, which in turn requires a number of *components* to be implemented.

In the above approach, *decisions* provide a simplified high-level map onto *features* and can be used to abstract from details by asking a few major questions that are relevant for a particular stakeholder. A *component model* describes components that implement the *features*. Making a *decision* can involve the selection of multiple *features*, each of which may require or exclude sets of other *features*. These *features* in turn may require or exclude sets of *components*. Furthermore, a relationship itself between two *features* may be implemented by a *component*. More details of the underlying model are described in [7].

Visualization of the relationships within a *feature model* alone is challenging, and numerous approaches have been proposed, ranging from filtered lists (e.g., [6]) to graph-based views (e.g., [12]) to methods of only showing the relationships on demand (e.g., [7]). With multiple models in place, visualizing the relationships between each of them becomes even more difficult. Presentation and manipulation of the underlying data in the execution of specific tasks is impeded by their multilayered interrelationships. For example, as mentioned above, making a *decision* can involve the selection of multiple *features*, each of which may require or exclude sets of other *features* and *components*. In such scenarios, stakeholders need to be presented with the relevant data using appropriate techniques. This will enable them to understand the current state and the impact of various required changes. Stakeholders also need to be able to make such changes with ease.

3.3 Task Support

The task of configuring a complete *feature model* can be reduced to a sequence of configuration decisions on individual features. At a basic level, this involves the ability to either select or eliminate a *feature* from the product under derivation which, in turn, usually leads to the inclusion, exclusion, or configuration of related *components*. Additionally, the ability to select or eliminate *features* in groups based on higher-level requirements (*decisions*) is a fundamental task. Whereas these tasks may seem basic, it is the knowledge and understanding (cognition) of the stakeholder that allows these tasks to be performed correctly. Drawing on a variety of research that has been carried out (e.g., [1, 2, 8]), we outline a set of simple cognitive tasks that aim to support the activity of the primary task—namely, to decide which *features* should be included and which should be excluded:

1. Identify/locate a configuration *decision*
2. Understand the high-level impact of a *decision* inclusion (perception of scale and nature of the impact—implements/requires/excludes)
3. Identify/locate a specific *feature*
4. Identify a specific *feature's* context—parent *feature*, alternative/supporting *features*, and *sub-features*
5. Understand the high-level impact of a *feature* selection—a specific *feature's* constraints (requires/excludes relationships)
6. Identify the state of a *feature*—selected/eliminated and why

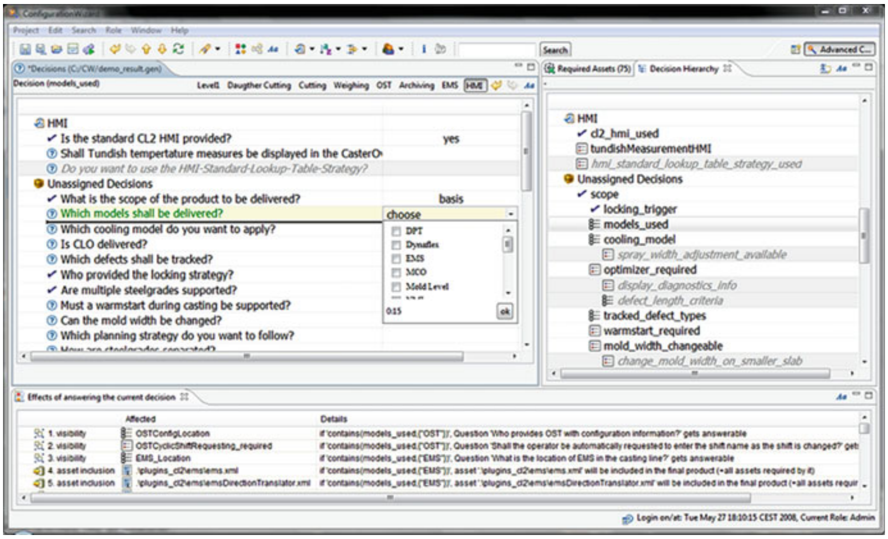
Visualization approaches can support these cognitive tasks by providing an interactive visual environment.

4 Visual Approaches and Implementations

As discussed, the comprehension and management of large sets of complex data relationships is the primary challenge when presenting variability data. Most approaches to date have utilized existing and well-known visual forms familiar to the software engineering community. The most prevalent of these is the ubiquitous "file explorer style" tree generally presented in the form of a static horizontal tree with expandable and collapsible branches. Recent work such as [13] has expanded on this visual form by introducing more dynamic tree structures and layouts. Other work (e.g., [14]) has focused on leveraging various techniques from the visualization community and utilizing alternative approaches not yet explored for this purpose.

When using visualization techniques for the handling and configuration of variability models, we have to address the cognitive tasks discussed earlier (see list in Sect. 3.3) with corresponding visual and interactive techniques. For instance, a tool environment has to provide interactive techniques to locate a *feature*, to understand a *feature's* configuration state, or to understand the impact of making a configuration decision.

Here, the resulting challenges are mostly related to the complexity and the scale of the models. In other words, the visualization and interactive technique must allow the stakeholder to handle large models and to focus on the relevant information, while abstracting from irrelevant details. This can be, for instance, achieved by techniques that allow to navigate on large models and to focus on elements for a particular task (e.g., a set of currently focused features) and related information (e.g., other elements in the model related to this feature set). A related challenge is that there are multiple ways to structure a model (e.g., which hierarchy to choose, how to modularize) and that the information structure that would be optimal for a particular task is not necessarily identical to the main structure of the model. Since a model is used for multiple tasks, visualizations and interactive techniques have to provide a means to adapt to different usage contexts and to change the focused aspects.


Project Edit Search Role Window Help
Search
Advanced C...
*Decisions (C:/CW/demo_result.gen)
Decision (models_used)
Level1 Daugther Cutting Cutting Weighing OST Archiving EMS HMI
Required Assets (75)
Decision Hierarchy
Effects of answering the current decision

HMI
- ✓ Is the standard CL2 HMI provided? yes
- Shall Tundish tempertature measures be displayed in the CasterO
- *Do you want to use the HMI-Standard-Lookup-Table-Strategy?*

Unassigned Decisions
- ✓ What is the scope of the product to be delivered? basis
- Which models shall be delivered? choose
- Which cooling model do you want to apply?
- Is CLO delivered?
- Which defects shall be tracked?
- ✓ Who provided the locking strategy?
- ✓ Are multiple steelgrades supported?
- Must a warmstart during casting be supported?
- Can the mold width be changed?
- Which planning strategy do you want to follow?

☐ DPT
☐ Dynaflex
☐ EMS
☐ MCO
☐ Mold Level
0:15 ok

HMI
- ✓ cl2_hmi_used
- tundishMeasurementHMI
- *hmi_standard_lookup_table_strategy_used*

Unassigned Decisions
- ✓ scope
  - ✓ locking_trigger
  - models_used
  - cooling_model
    - *spray_width_adjustment_available*
  - optimizer_required
    - *display_diagnostics_info*
    - *defect_length_criteria*
  - tracked_defect_types
  - warmstart_required
  - mold_width_changeable
    - *change_mold_width_on_smaller_slab*

	Affected	Details
1. visibility	OSTConfigLocation	if 'contains(models_used,{"OST"})', Question 'Who provides OST with configuration information?' gets answerable
2. visibility	OSTCyclicShiftRequesting_required	if 'contains(models_used,{"OST"})', Question 'Shall the operator be automatically requested to enter the shift name as the shift is changed?' get
3. visibility	EMS_Location	if 'contains(models_used,{"EMS"})', Question 'What is the location of EMS in the casting line?' gets answerable
4. asset inclusion	.\plugins_cl2\ems\ems.xml	if 'contains(models_used,{"EMS"})', asset '.\plugins_cl2\ems\ems.xml' will be included in the final product (+all assets required by it)
5. asset inclusion	.\plugins_cl2\ems\emsDirectionTranslator.xml	if 'contains(models_used,{"EMS"})', asset '.\plugins_cl2\ems\emsDirectionTranslator.xml' will be included in the final product (+all assets requir

Login on/at: Tue May 27 18:10:15 CEST 2008, Current Role: Admin

Fig. 7.2 DOPLER configuration wizard [6]

We will now look at particular approaches in more detail. In general, the approaches and techniques to variability visualization can be divided into three broad areas: two dimensional (2D), two and a half dimensional (2.5D), and three dimensional (3D). General characteristics of these approaches and implementation examples from a number of tools supporting variability management are described in the following subsections. The examples are taken from both research and commercially available tool suites.

4.1 2D Visualization

Using 2D approaches such as matrices and graphs to visualize feature models is the normative way to allow feature exploration and model manipulation [6, 8]. More recently, research tools are exploring the use of alternative tree layouts such as dynamic space trees [9] and radial trees [13]. In conjunction with this, the use of varying visualization techniques as described in Sect. 2 is being employed to aid stakeholder cognition in variability management tasks.

4.1.1 Examples

The *DOPLER* tool suite [6] provides decision-oriented variability management through a number of complimentary tools. One of these tools, the *Configuration Wizard*, provides capabilities for product customization, requirements elicitation, and configuration generation. Figure 7.2 shows the use of hierarchical tree structures to display a set of decisions on the left and the decision hierarchy on

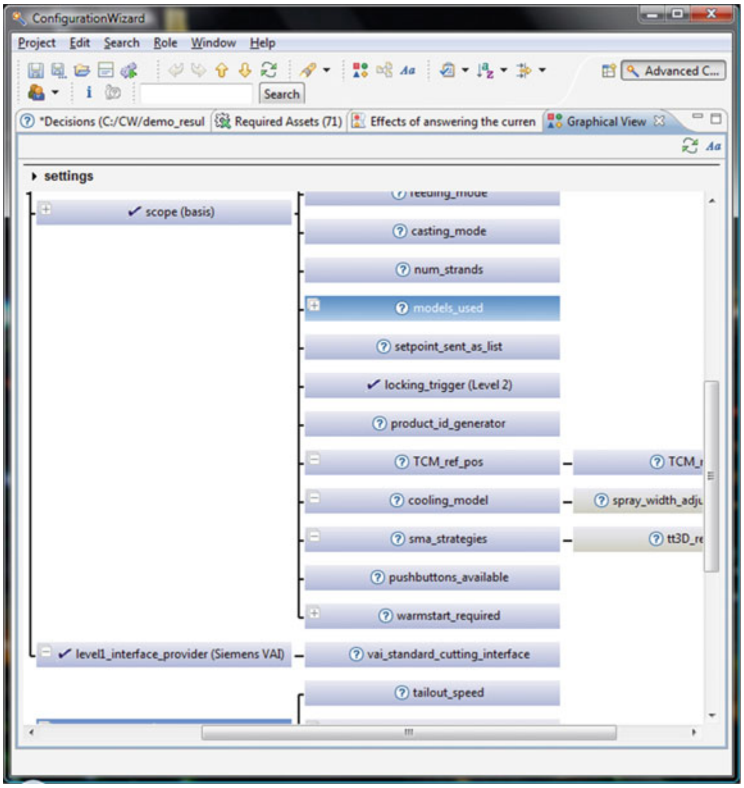


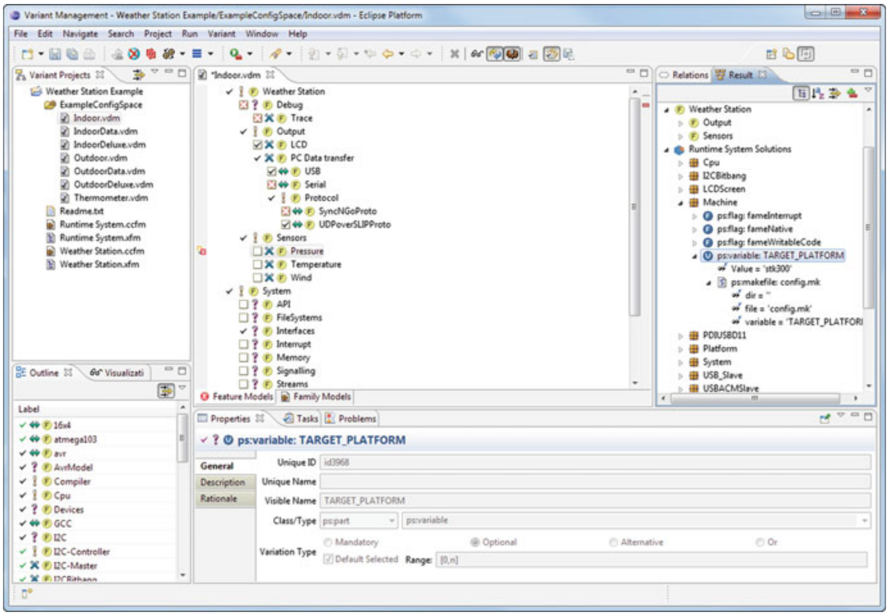
Fig. 7.3 DOPLER tree view [29]

the right. Figure 7.3 shows a more graphical representation of the tree structure. Both visualizations of the model utilize simple iconography to encode selected items (tick symbol icon) and items not yet configured (question mark icon).

The *pure::variants* tool suite [15, 16] is a commercially available product, which provides a set of integrated tools that support various phases of the software product line development and derivation process.

For creating and configuring a new software variant, the tool provides a *Configuration Editor* (see Fig. 7.4). This editor employs a hierarchical "file explorer style" horizontal tree to allow the browsing, selection, and de-selection of features according to their constraints. Iconography is extensively used to identify element types and feature state. Figure 7.5 shows a matrix visualization, which presents a view of the variants identifying the different features available in each of the variants.

The research tool *vivid* [14] primarily explores the use of more dynamic and interactive visualizations in order to provide cognitive support to stakeholders. Figure 7.6 shows a horizontal tree visualization, which represents a variant's *feature* configuration. The visualization allows the stakeholder to incrementally browse the tree structure automatically collapsing and expanding relevant branches

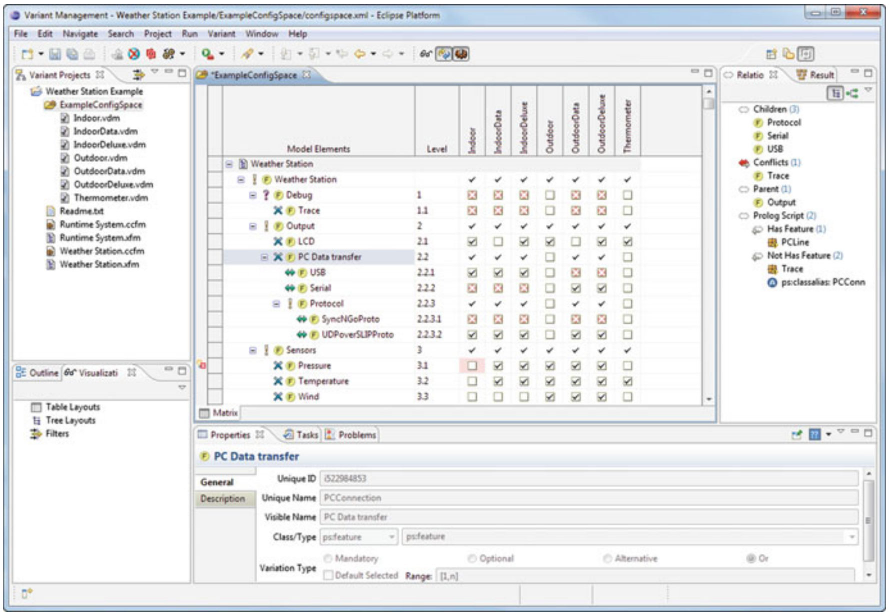

Variant Management - Weather Station Example/ExampleConfigSpace/Indoor.vdm - Eclipse Platform
File Edit Navigate Search Project Run Variant Window Help
Variant Projects
*Indoor.vdm
Relations
Result
Outline
Visualizati
Feature Models
Family Models
Properties
Tasks
Problems

✓ ? ps:variable: TARGET_PLATFORM

General | Description | Rationale

Unique ID: id3968
Unique Name:
Visible Name: TARGET_PLATFORM
Class/Type: ps:part | ps:variable
Variation Type: ( ) Mandatory [✓ Optional] ( ) Alternative ( ) Or
[✓ Default Selected] Range: [0,n]

Fig. 7.4 pure::variants configuration editor [16]


Variant Management - Weather Station Example/ExampleConfigSpace/configspace.xml - Eclipse Platform
File Edit Navigate Search Project Run Variant Window Help
Variant Projects
*ExampleConfigSpace
Relatio
Result
Outline
Visualizati
Matrix
Properties
Tasks
Problems

Model Elements	Level	Indoor	IndoorData	IndoorDeluxe	Outdoor	OutdoorData	OutdoorDeluxe	Thermometer
Weather Station								
Weather Station		✓	✓	✓	✓	✓	✓	✓
Debug	1	☒	☒	☒	☐	☒	☒	☐
Trace	1.1	☒	☒	☒	☐	☒	☒	☐
Output	2	✓	✓	✓	✓	✓	✓	✓
LCD	2.1	☑	☐	☑	☑	☐	☑	☑
PC Data transfer	2.2	✓	✓	✓	☐	✓	✓	☐
USB	2.2.1	☑	☑	☑	☐	☒	☒	☐
Serial	2.2.2	☒	☒	☒	☐	☑	☑	☐
Protocol	2.2.3	✓	✓	✓	☐	✓	✓	☐
SyncNGoProto	2.2.3.1	☒	☒	☒	☐	☒	☒	☐
UDPoverSLIPProto	2.2.3.2	☑	☑	☑	☐	☑	☑	☐
Sensors	3	✓	✓	✓	✓	✓	✓	✓
Pressure	3.1	☐	☑	☑	☑	☑	☑	☐
Temperature	3.2	☐	☑	☑	☑	☑	☑	☑
Wind	3.3	☐	☐	☐	☑	☑	☑	☐

PC Data transfer

General | Description

Unique ID: i522984853
Unique Name: PCConnection
Visible Name: PC Data transfer
Class/Type: ps:feature | ps:feature
Variation Type: ( ) Mandatory ( ) Optional ( ) Alternative [✓ Or]
[ ] Default Selected Range: [1,n]

Fig. 7.5 pure::variants matrix view [16]

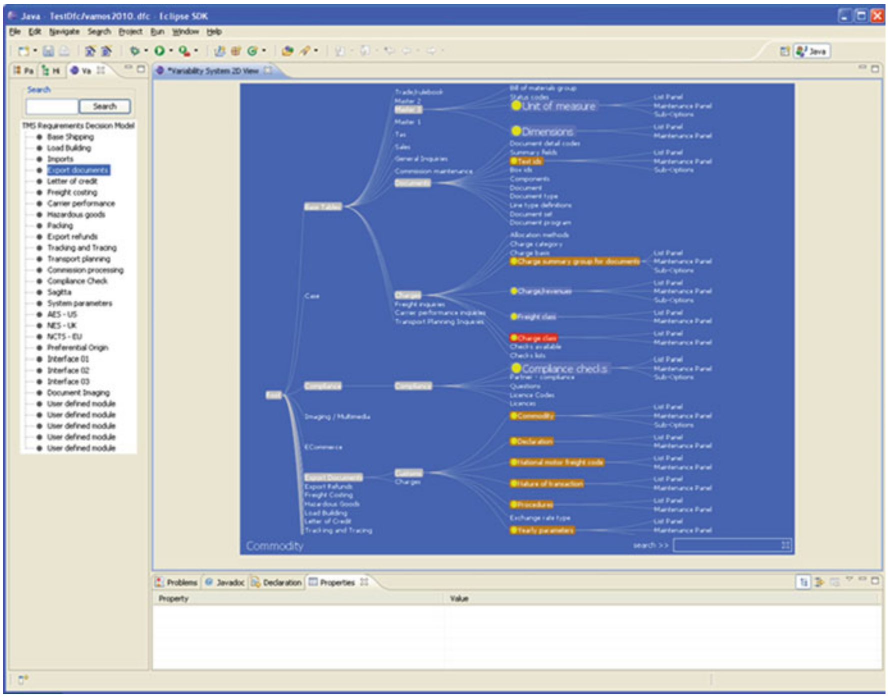


Fig. 7.6 vivid tree view [14]

as the stakeholder progresses. Animation of any visual changes to the display aids the stakeholder in understanding the navigation path. Color encoding allows immediate identification of the state of a particular feature.

The feature configuration tool *S2T2 Configurator* developed in earlier work by Botterweck et al. [17] provides techniques for the configuration of complex feature models and techniques for the joint visualization of feature and implementation models. In [18] the approach was extended by "Feature Flow Maps" to visualize product attributes, which result from configuration decisions, during product configuration (see Fig. 7.7). For instance, the width of the lines indicates the price of the product resulting from the current feature configuration. This visualization is updated incrementally while the feature configuration process is completed.

4.1.2 Advantages and Limitations

The advantages of using visual representations such as lists, "file explorer style" trees, and matrix tables are evident—they are generally familiar and intuitive to stakeholders. However, when the variability that exists within a software product line contains thousands of variation points, it becomes difficult to manage and cognitively challenging to navigate.

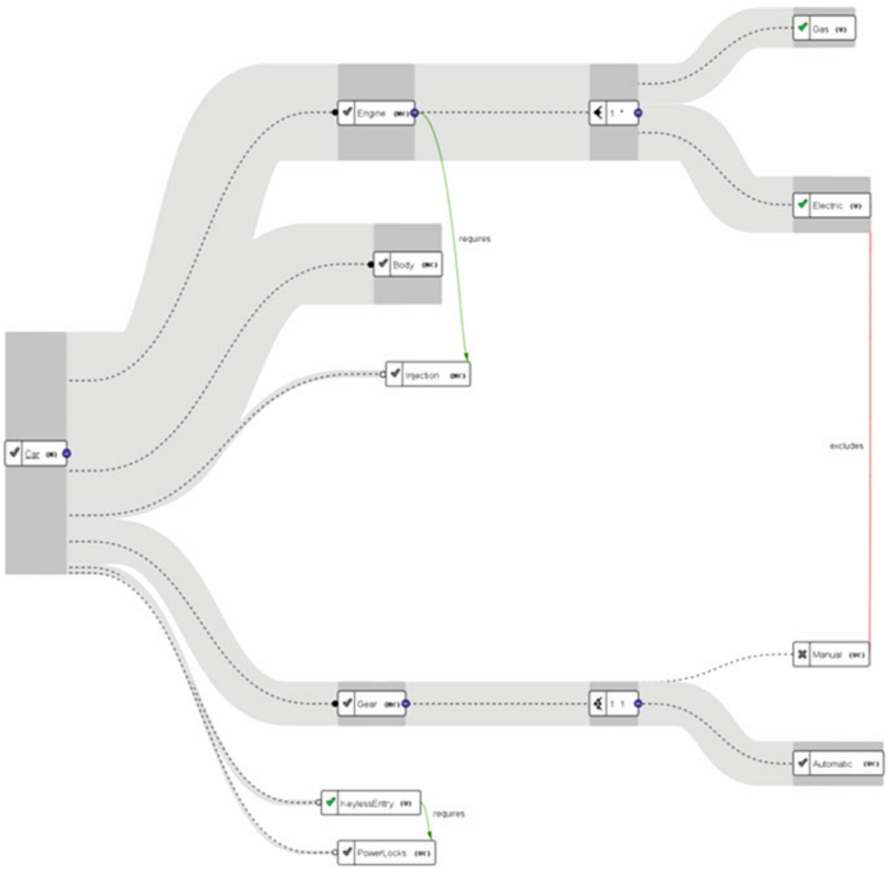


Fig. 7.7 Feature flow maps [18]

Using dynamic tree structures (as with *vivid*) and techniques such as animation, degree of interest, and distortion, visualizations can provide cognitive support to aid with such challenges. However, even with such additional aids, it can still be challenging to configure variability for very large product lines. In the next two sections, we show some examples of alternative visualizations being explored to further enhance the cognitive support provided to stakeholders.

4.2 2.5D Visualization

2.5D visualization techniques use 3D visual attributes in a 2D display [19]. For example, adding 3D attributes such as perspective (e.g., making certain objects smaller to indicate distance) and occlusion (e.g., overlapping objects to indicate layers) to a 2D display can be described as creating a 2.5D display. Work into the

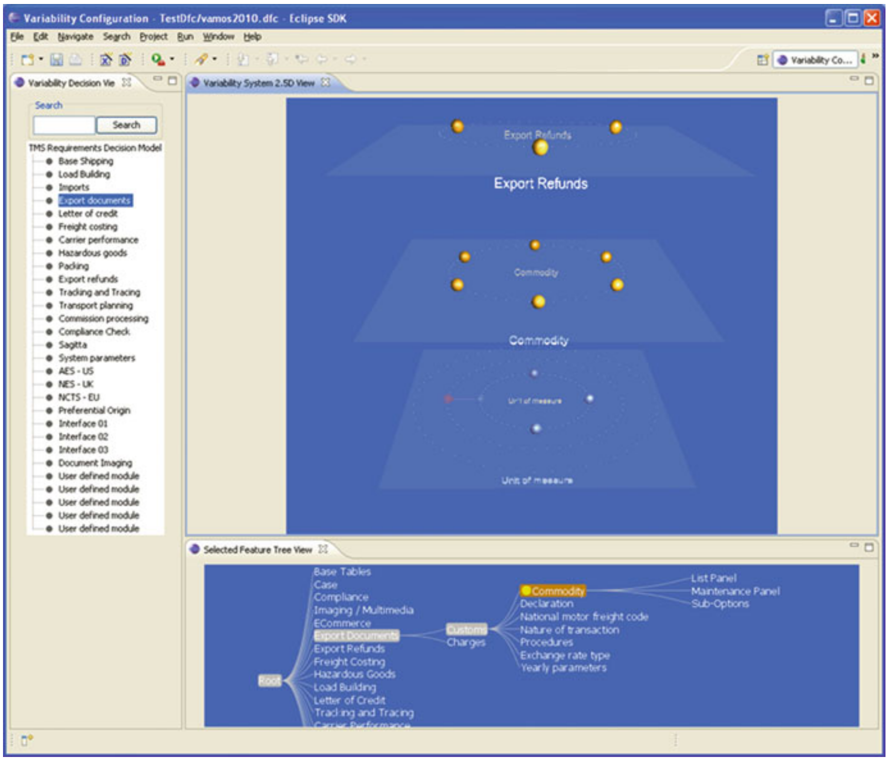


Fig. 7.8 vivid 2.5D view [14]

employment of such techniques uses static 3D planes on which representations of features are presented in an animated interactive environment.

4.2.1 Example

Figure 7.8 shows a 2.5D visualization from the *vivid* prototype. To the left is a simple list of *decisions* (high-level grouping of features). When a selection is made within this supporting view, the main view displays the implementing features along with all features that are required or excluded by them.

The view consists of three stacked planes. Each plane provides a circular grouping of spheres. In the top plane, each sphere in the circle represents a grouping of features. When any one of those groupings in the top plane is selected (by mouse click), then all features that comprise that grouping are displayed in the middle plane in a similar circular format. In the lower plane, all related (required/excluded) features are displayed (for all features presented in the middle plane). The innermost circle on the lower plane identifies features that are directly related (required, excluded) to features in the middle plane. In order of ascending radii, each subsequent circle in the lower plane represents the transitive relationships that exist; i.e., required features can further require and/or exclude other features.

By hovering the mouse over any sphere in any of the planes, a description of that element will be displayed in the center of the plane. When a sphere is selected in any plane, the circle on which it is presented will rotate so that sphere is brought to the front with its description displayed underneath. These functions aim to implement *Details on Demand*.

Each sphere acts as a representation of a specific feature. A sphere may be color encoded to visually identify its relationship to other variability artifacts (the feature implements a decision or the feature is required/excluded by another feature).

Multiple windows (and multiple planes) are employed to separate and distribute decisions, feature groupings, features, and relationships. Note that the lower plane displays all related features for all the implementing features in the middle plane. This allows an overview of the impact as a whole for this group of features. When a single implementing feature is selected in the middle plane, the circles in the lower plane rotate to ensure all related features are brought to the front while all other features in the plane are distorted (made transparent) in order to highlight the ones of interest. Animation is again used for all movements to preserve context.

4.2.2 Advantages and Limitations

The primary aim of the 2.5D approach is to increase the number of features and relationships that can be represented at any given time within a fixed screen space avoiding the need of panning and zooming across thousands of related on-screen elements. This is achieved by utilizing a depth dimension and providing animated movement and highlighting of relevant information to the foreground when required. *Focus+Context* is a key consideration here.

The presence of a fixed on-screen space may reveal a limitation of this visualization: there will be a point where a very large number of features and relationships may cause unwanted occlusion and selection difficulties. However, this situation would only occur with extremely large and/or complex feature models. Testing and usability studies are required to evaluate the effectiveness of this approach.

4.3 3D Visualization

Differing reports exist on the effectiveness of 3D visualizations to support software engineering but literature suggests that there is acceptance that it can be effective in specific instances (e.g., [20, 21]). Current work into the use of 3D primarily focuses on the visualization of relationships instead of the elements that they relate [9]. Relationships are visualized as objects in a 3D space whose coordinates identify elements within three different models, each model being mapped to one of the three axes.

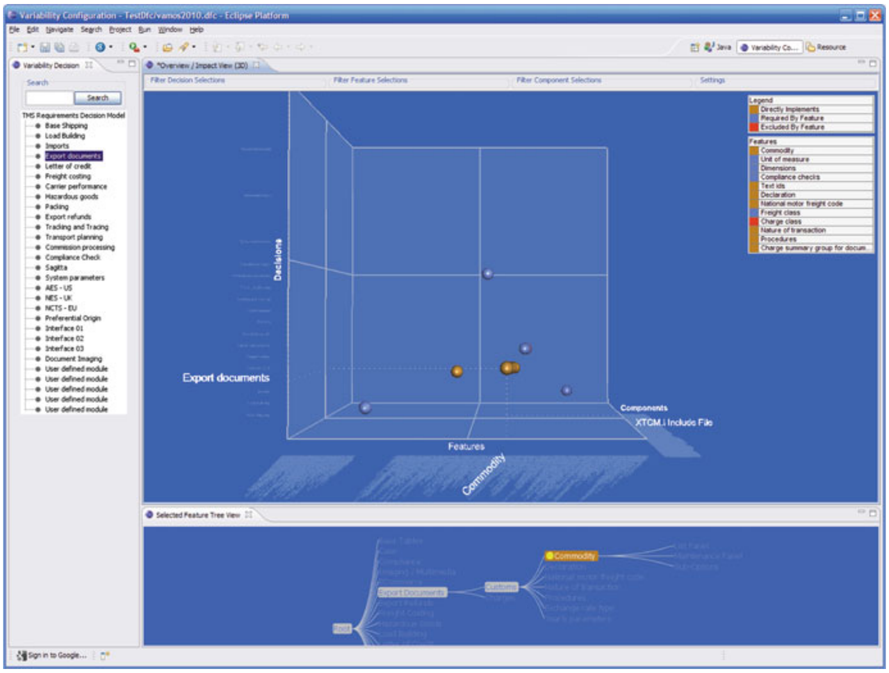


Fig. 7.9 vivid 3D view [14]

4.3.1 Example

Figure 7.9 presents a 3D view which attempts to provide a self-contained representation of all the three models introduced in Sect. 3.2 (*decisions*, *features*, and *components*) and their interrelationships. However, at any given time, only information of interest is displayed.

Here, as in the 2.5D approach, multiple windows are employed to distribute the information and provide the supporting *decision* view. The visualization consists of a 3D space containing X, Y, and Z axes. Sequential lists of *decisions, features*, and *components* are displayed along the Y-axis, X-axis, and Z-axis (moving away from the observer), respectively.

The key idea here is that a point within this 3D space identifies a relationship between all three models. In other words, a sphere plotted at a particular point will identify that the *feature* labeled at its X-coordinate implements the *decision* labeled at its Y-coordinate and is implemented by the *component* labeled at its Z-coordinate. In Fig. 7.9, the stakeholder has highlighted the sphere that represents the "Commodities" *feature*. However, in addition to this, by looking at the highlighted labels on the axes, we can see that it also represents the "Export Documents" *decision* that the *feature* implements and the "XTCM.I Include File" component that implements the *feature*.

Focus+Context and *Details on Demand* are the main techniques guiding this implementation. One goal is that all three models can be perceived to be represented through the listings on each axis. However, the details of any part of any model or its relationships are only displayed when required. For example, when a *decision* is selected, there may be a number of implementing *features*. For each implementing *feature*, a sphere is plotted in the 3D space as described above. Other *features* required or excluded by those implementing features are similarly plotted as spheres and are given a specific color encoding which allows a visual identification of the required or excluded relationship.

Pan & Zoom are combined with rotation to allow a full *world-in-hand* manipulation of the view in three dimensions allowing the stakeholder to position the view depending on the information of interest.

4.3.2 Advantages and Limitations

One advantage argued with this visualization is that it provides a perception of a software product line as a whole within a 3D configuration space while only presenting data that is relevant at a given time. Visually, a stakeholder is enabled to comprehend both the scale and nature of selecting a *decision*, *feature*, or *component*. As such, selecting a *decision* for implementation will require a set of implementing *features* but also require and exclude a large set of other *features*. The impact of such a *decision*, including its nature and magnitude, will be immediately evident allowing the stakeholder to further investigate the details of the impact.

As with the 2.5D visualization, the fixed on-screen space within a 3D configuration may also be a limitation as there is a point at which a very large number of features and relationships will cause unwanted occlusion and selection difficulties. However, this situation would only occur with extremely large and/or complex feature models. Again, testing and usability studies are required to evaluate the effectiveness of this approach.

5 Related Work

Traditionally, many approaches that support feature configuration as part of product line derivation use a hierarchical model. The visualization of hierarchical structures has been studied extensively in the visualization literature, including node-link techniques (e.g., [22]), space-filling techniques (e.g., Tree Maps [23]), and interactive techniques that help to cope with very large models such as *Focus+Context* (e.g., [24]).

Approaches focusing on multiple hierarchies are useful when visualizing the relationships between features and other models as discussed above. Robertson et al., for example, define polyarchies [25] as multiple hierarchies that share nodes.

They describe the visualization design and a software architecture for displaying polyarchy data from a set of hierarchical databases. They use animated transitions when switching between the related hierarchies to allow the user to keep context. Polyarchies are somewhat similar to the multiple related hierarchies found in some product line configuration approaches but lack the intra-model relations and the aspect of progressing configuration.

Moreover, a number of research tools which provide product line configuration capabilities and apply visualization techniques exist in literature. Some of these tools have already been discussed in the preceding section (e.g., [6, 14, 16, 18]). Another example is the research prototype *V-Visualize* developed by Sellier and Mannion [26], which visualizes configuration decisions with a force-directed layout.

Some approaches aim to visualize the variability in the artifacts which are influenced by this variability. For instance, Kästner et al. [27] used color encoding to indicate variability in program code.

In an earlier work, the authors presented Visit-FC, a configuration approach and tool that indicates the configuration state of features by visual clues [28].

6 Summary

The mechanisms by which software variability is presented to a stakeholder through visual representation and interactivity can have a substantial effect on how efficiently the stakeholder can perform their required management tasks. This becomes more and more evident as the size and complexity of the variability increases. Many of today's variability management tools use normative software engineering user interface techniques to present, and provide management of, that variability. For example, variability artifacts such as *decisions*, *features*, and *components* can be presented in one-dimensional lists, as elements in a two-dimensional matrix/spreadsheet, or as nodes in "file explorer" style trees that provide grouping and allow selection/elimination from a variant model.

As industrial-sized product lines grow to the order of many thousands of variation points, these traditional techniques tend to be limited in the cognitive support that they provide to stakeholders in relation to the performance of their tasks. Information visualization techniques have proven useful in enhancing cognition in numerous ways, and in recent work, these techniques are being employed with the aim of increasing the cognitive support provided. Visualization strategies such as *Focus+Context*, *Degree of Interest*, and *Details on Demand* in combination with techniques such as *Iconography*, *Color Encoding*, and *Distortion* are being utilized leveraging the work that has been carried out within the information visualization community. Implementations using dynamic space trees, radial graphs, and more explorative 2.5D and 3D techniques have been developed. Table 7.1 briefly summarizes advantages and limitations of approaches discussed in this chapter.

Table 7.1 Overview of approaches

	Example approaches	Advantages	Limitations
2D	DOPLER [6] pure::variants [15] S2T2 Configurator [18] vivid tree view [14]	"Explorer" interaction style is familiar to stakeholders	Limited on-screen space
2.5D	vivid 2.5D view [14]	Representation of a larger number of elements (features) in a limited space	Interaction requires training
3D	vivid 3D view [14]	Perception of product line as a whole Natural representation of scale	Interaction requires training

7 Outlook

There are a variety of commercial and research tools that provide support for variability management. Many of these are continuing in their development and some are actively exploring the use of more novel presentation and interactive techniques to improve their support for small- and large-scale variability projects. In the immediate future, the use of dynamic graphs and, in particular, the use of *degree of interest* trees seems to become more prevalent. There is also ongoing work into the use of 2.5D and 3D strategies which aims to leverage more of the visualization research that has been carried out to date.

References

1. Deelstra, S., Sinnema, M., Bosch, J.: Product derivation in software product families: a case study. J. Syst. Softw. **74**, 173–194 (2005)
2. Steger, M., Tischer, C., Boss, B., Müller, A., Pertler, O., Stolz, W., Ferber, S.: Introducing PLA at Bosch Gasoline Systems: experiences and practices. In: SPLC 2004, Boston, MA, pp. 34–50 (2004)
3. Card, S.K., Mackinlay, J.D., Shneiderman, B.: Readings in Information Visualisation: Using Vision to Think. Morgan Kaufmann, San Francisco, CA (1999)
4. Ware, C.: Information Visualisation: Perception for Design, 2nd edn. Morgan Kaufmann, San Francisco, CA (2004)
5. Diehl, S.: Software Visualization – Visualizing the Structure, Behaviour, and Evolution of Software. Springer, Heidelberg (2007)
6. Rabiser, R., Dhungana, D., Grünbacher, P.: Tool support for product derivation in large-scale product lines: a wizard-based approach. Presented at the 1st International Workshop on Visualisation in Software Product Line Engineering (ViSPLE 2007), Tokyo, Japan (2007)
7. Botterweck, G., Thiel, S., Nestor, D., Abid, S.B., Cawley, C.: Visual tool support for configuring and understanding software product lines. Presented at the 12th International Software Product Line Conference (SPLC08), Limerick, Ireland (2008)
8. Sinnema, M., Graaf, O. d., Bosch, J.: Tool support for COVAMOF. Presented at the Workshop on Software Variability Management for Product Derivation – Towards Tool Support (2004)

9. Cawley, C., Healy, P., Thiel, S., Botterweck, G.: Research tool to support feature configuration in software product lines. Presented at the 4th International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS) Linz, Austria (2010)
10. Czarnecki, K., Helsen, S., Eisenecker, U.: Staged configuration using feature models. Presented at the Proceedings of the Third Software Product Line Conference, Boston, MA (2004)
11. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, S.: Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University (1990)
12. Sellier, D., Mannion, M.: Visualizing product line requirement selection decisions. Presented at the 1st International Workshop on Visualisation in Software Product Line Engineering (ViSPLE 2007), Tokyo, Japan (2007)
13. Rabiser, R.: Flexible and user-centered visualization support for product derivation. Presented at the 2nd International Workshop on Visualisation in Software Product Line Engineering (ViSPLE), Limerick, Ireland (2008)
14. Cawley, C., Healy, P., Botterweck, G.: A discussion of three visualisation approaches to providing cognitive support in variability management. Presented at the 2nd Conference on Software Technologies and Processes (STeP), Furtwangen, Germany (2010)
15. Beuche, D.: Modeling and building software product lines with pure::variants. In: 12th International Software Product Line Conference (SPLC 2008), Limerick, Ireland (2008)
16. pure-systems GmbH. Variant management with pure::variants. pure-systems GmbH (2006)
17. Botterweck, G., Janota, M., Schneeweiss, D.: A design of a configurable feature model configurator. In: Proceedings of the 3rd International Workshop on Variability Modelling of Software-Intensive Systems (VAMOS 09), pp. 165–168 (2009)
18. Schneeweiss, D., Botterweck, G.: Using flow maps to visualize product attributes during feature configuration. In: VISPLE 2010, Jeju Island, Korea (2010)
19. Ware, C.: Designing with a 2 1/2D attitude. Inf. Des. J. **3**, 255–262 (2001)
20. Ali, J.: Cognitive support through visualization and focus specification for understanding large class libraries. J. Vis. Lang. Comput. **20**(1), 50–59 (2009)
21. Risden, K., Czerwinski, M.P., Munzner, T., Cook, D.B.: An initial examination of ease of use for 2D and 3D information visualizations of web content. Int. J. Hum. Comput. Stud. **53**(5), 695–714 (2000)
22. Walker, J.Q.: A node-positioning algorithm for general trees. Softw. Pract. Exp. **20**, 685–705 (1990)
23. Shneiderman, B.: Tree visualization with tree-maps: 2-d space-filling approach. ACM Trans. Graph. **11**, 92–99 (1992)
24. Cockburn, A., Karlson, A., Bederson, B.B.: A review of overview+detail, zooming, and focus +context interfaces. ACM Comput. Surv. **41**, 1–31 (2008)
25. Robertson, G., Cameron, K., Czerwinski, M., Robbins, D.: Polyarchy visualization: visualizing multiple intersecting hierarchies. In: ACM CHI 2002 Conference on Human Factors in Computing Systems, pp. 423–430 (2002)
26. Sellier, D., Mannion, M.: Visualizing product line requirement selection decisions. In: SPLC (2), pp. 109–118 (2007)
27. Kästner, C., Trujillo, S., Apel, S.: Visualizing software product line variabilities in source code. Presented at the VISPLE 2008, Limerick, Ireland (2008)
28. Botterweck, G., Thiel, S., Cawley, C., Nestor, D., Preussner, A.: Visual configuration in automotive software product lines. In: 2nd IEEE International Workshop on Software Engineering Challenges in Automotive Domain (SECAD 2008), held in conjunction with IEEE COMPSAC 2008, Turku, Finland (2008)
29. Rabiser, R.: Flexible and user-centered visualization support for product derivation. In: Proceedings of the 12th International Software Product Line Conference (SPLC 2008), Second Volume, 2nd International Workshop on Visualisation in Software Product Line Engineering (ViSPLE 2008), Limerick, Ireland, pp. 323–328. Lero (2008)