Visualising Inter-Model Relationships in Software Product Lines

Ciarán Cawley, Steffen Thiel, Goetz Botterweck, Daren Nestor

Lero, University of Limerick
Limerick, Ireland
{ ciaran.cawley | steffen.thiel | goetz.botterweck | daren.nestor }@lero.ie

Abstract
Within Software Product Line (SPL) Engineering, Feature modelling is a prevalent mechanism for managing variability but is insufficient for describing it as a whole and for relating its different aspects. Other modelling techniques such as Decision modelling and Component modelling provide different views of the underlying SPL data. To facilitate certain approaches in product line engineering, such as tool-supported product derivation or automated analyses we need to describe the SPL with multiple inter-related models. In this paper, we discuss how to represent the relationships that exist between those models and present an approach for communicating the relationships using visualisation techniques. We also discuss the visualisation through example scenarios and argue its benefits.

1. Introduction
Software Product Line (SPL) Engineering focuses on identifying and managing the commonalities and variability of a set of software products such that core assets can be developed and (re)used to derive individual product variants with a minimum of cost [1]. Within industry, software product lines exist with thousands of variation points presenting a scale of variability where comprehension and manageability become difficult to achieve [2], [3]. The management of such a product line’s variability is fundamentally key to its success. Particularly in the areas of modelling of the product line and product configuration, variability management can greatly impact the complexity that is involved when producing a new product from existing product line assets [4].

Visualisation is widely used in software engineering and has proven useful to amplify human cognition in data intensive applications. It takes abstract data and transforms it into a format that is presentable to humans. By leveraging the principles and techniques developed by visualisation communities [5] we can attempt to address the scale and complexity issues that challenge large scale software product line efficiency.

Feature modelling is currently a prominent mechanism for managing software product line commonality and variability e.g. [6]. However, the authors will argue that a feature model in itself does not adequately describe a product line. Other mechanisms for managing the complexity of a product line, such as decision modelling [7] and component modelling [8] endeavour to present a different view of the underlying data that comprises a software product line. In this paper, we present a visualisation approach that attempts to address the challenges of representing multiple models and their relationships with each other.

In Section 2, we motivate the need for multiple models and why they need to be integrated. In Section 3 we discuss the challenges faced by this integration. In Section 4 we present our visualisation approach and in Section 5 summarise a tool framework that will incorporate it. Section 6 presents related work and Sections 7 and 8 outline future work and conclude the paper respectively.

2. Motivation
Most SPL research approaches focus on single development artefacts, represented by isolated models (e.g. [9] [10]). Commonly, these artefacts are features. While viewing a product line as a collection of features has many advantages, there are some disadvantages also. Some of the disadvantages include the problem of describing cross-cutting features, describing non-functional requirements, and the problems that arise in linking a feature to a concrete component (or set of components) that implement that feature.

There are numerous tasks that need to be performed by various stakeholders during the SPL engineering processes of domain engineering (development of core assets) and application engineering (specific product
variant derivation). Platform managers, domain engineers, product managers, application engineers, customers and developers all perform 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 focussed manner and supports stakeholder and task specific representation and manipulation. Section 3 discusses the resulting challenges that arise, namely the need to support the representation and affects of the relationships that exist once separate models are employed.

3. Inter-Model Relationships

The approach presented in this paper is based on the meta-models presented in [6]. These meta-models are not described here however the interested reader is directed to the relevant reference.

To exploit the benefits of a product line we need to connect the isolated models that describe the individual portions of the product line such as decision models, feature models and component models. In our approach, decisions provide a simplified high-level view onto features and can be used to abstract from details by asking a few major questions which are relevant for a particular stakeholder. A component model describes components that implement 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.

Visualisation of the relationships within a feature model alone is challenging, and numerous approaches have been proposed, from filtered lists [7] to graph based views [11] to methods of only showing the relationships on demand [12]. With multiple models in place, visualising 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 the multilayered relationships that exist between them. 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. These features in turn may require or exclude sets of components. In scenarios like this, stakeholders need to be presented with the relevant data using appropriate techniques that allow them to understand the current state and the impact of various changes that may be necessary. They also need to be able to make those changes easily. Crosscutting requirements are another major challenge. One way of describing a crosscutting requirement or feature is through use of aspects.

4. Relationship Visualisation Approach

This section presents an approach to visualising relationships that exist between the various models. The primary concept is to focus the attention of the visualisation on the relationships that exist between elements of one model and elements of other models (e.g. feature implemented by component). This is in contrast to traditional approaches which focus primarily on the elements (such as features or components) of the models. Such approaches subsequently describe how those elements are connected (such as node-link diagrams) [6], [11]. In this approach, the relationships themselves are the prominent visual element in the view. Using this concept, the visualisation aims to provide a high-level context of relationships relevant to given stakeholder tasks along with the ability to focus on specific data elements that support those tasks (examples are discussed later in this section). The ability of a stakeholder to interact, explore and query the visualisation is of primary importance. The following subsections explain and illustrate this approach providing example scenarios. Subsection 4.1 introduces the concept of a relation point while subsection 4.2 elaborates on its use within our approach.

4.1. Relation Points

Figure 1 illustrates a 3D space. Each axis in this 3D space represents a different model. Features are represented along the X-axis, decisions are represented along a vertical Z-axis and components along the Y-axis. A relation point is a point plotted in the 3D space where a decision, feature and component intersect, i.e. where a decision is implemented by a feature and a feature is implemented by a component. These two inter-model relationships connecting three separate models are represented by a single visual element.

Figure 1 serves to illustrate a relation point with a simplified example. In this example, the decision to include Hardware Platform B (vertical Z-axis) results in the spherical relation point being displayed. (For the purposes of this paper, it is suggested that decisions are made through an associated synchronised view such as a decision list.) Three functions are evident. Firstly, the existence of the relation point identifies that relationships are impacted within the system. Secondly, with simple interaction such as clicking on the relation point, the stakeholder is informed that the Software Upgradeability feature has been selected and
that it is implemented by the Flash Rom component. This is shown by displaying / highlighting the decision, feature and component names on each of the axes. Thirdly, through the use of layering (tooltip style functionality), the two relationships in this example, decision is implemented by feature and feature is implemented by component, are displayed textually providing more detail on demand.

This single visual element can also represent many more relationships and attributes without adding additional visual clutter. Visual techniques such as colour encoding the relation point, use of varying iconography to render the relation point and/or animation of the relation point can be applied to represent additional dependencies. For example, the use of a cube instead of a sphere to represent the relation point shown in Figure 1 could indicate the feature is related to a fourth or fifth model such as a business driver or quality model. By querying the relation point as above, the details of the relationship can again be provided on demand. Other such models can also be mapped to axes and swapped in and out replacing existing model to axis mappings. The mapping of a model to an axis is a simple linear sequential listing of the model elements.

It is important to remember that a relation point represents relationships. As a consequence, for example, if a feature is implemented by two components then two different relation points will be displayed. One will represent the relationship feature implemented by component 1 and the other will represent feature implemented by component 2 and both will represent the same relationship decision implemented by feature. How this type of visualisation can aid a stakeholder using interaction is discussed in the following subsection.

Intra-model relationships are also encapsulated such as a green relation point indicating that the specific feature associated with the relation point is required due to an intra-model requires relationship with another feature. As briefly mentioned earlier, with a substantial amount of additional information available, specific data can be acquired on demand through stakeholder interaction such as mouse movements and events. The next section elaborates more on some of these points through an example and discusses the role of interaction within the visualisation.

**Figure 1. Relation Point Example**
4.2. An Interactive Derivation Space

Following the example presented in Figure 1, Figure 2 presents a less simplified illustration of the visualisation. Here a stakeholder has resolved two decisions, to include hardware platform B and hardware platform C. On the resolution (making) of these decisions, eleven relation points are visualised. A grey relation point indicates a decision implemented by feature relationship and a feature implemented by component relationship. With 6 grey relation points, 12 relationships in total are visualised. If a relation point is green it identifies an additional relationship which represents feature requires feature. If a relation point is red it identifies the additional relationship feature excludes feature. If a relation point is blue it identifies the additional relationship feature recommends feature. If a relation point is amber it identifies the additional relationship feature problematic with feature. In total, 27 relationships incorporating 3 separate models are visualised.

4.2.1. Stakeholder Interaction

As with any visualisation where a stakeholder is required to perform specific tasks, appropriate interaction is needed. We argue that this 3D visualisation allows for a substantial number of relation points to be rendered without causing information overload. This allows for a large amount of relational information to be presented. Further work is required to identify a balance between the number of relation points displayed based on specific tasks and the onset of such information overload.

3D visualisations afford the use of the world-in-hand metaphor where the stakeholder can rotate it as a whole in any direction. This kind of interaction allows a familiar exploration of the presented data where the stakeholder can manipulate the visualisation to gain an optimum view dependent on their interest. For example, where aspect oriented programming is used and in the event of say 50 relation points representing say 150 relationships, an application engineer can orient the visualisation so that the best possible view of all aspect implemented features is shown. Distortion techniques such as blurring and/or transparency could be applied by the application engineer to distort all non aspect implemented features thus highlighting what is of interest without losing the context of the rest of the relationships impacted by the decisions made.
As discussed in 4.1, relation points themselves can be interacted with through mouse movements and clicking. In this way details-on-demand can be rendered when required.

4.2.2. View Interaction

As was briefly discussed earlier, synchronising this visualisation with other views provides a number of additional functions. Providing a simple decision view such as a list allows a stakeholder to abstract details of the system at a high level. The approach presented in this paper can then act as an automatic rendering of the size and nature of the impact on the system as a whole providing immediate visual feedback through colour, shapes and iconography. Other views such as a traditional tree diagram [6] which provide contextual information for individual models can also be synchronised with our approach. For example, by clicking on a relation point, the nodes and links in a separate, associated tree view representing the relevant relationships between the different models can be highlighted. This affords a different perspective that may be more suited and functional for different stakeholder tasks.

4.3. Task Support

The primary aim of this visualisation is to provide cognitive support to various stakeholders which aids feature configuration during product derivation. The rest of this subsection discusses this in the context of the three models in the example.

When a stakeholder resolves a decision, this approach provides immediate visual feedback regarding the size and nature of the impact of that decision on the rest of the system. It does this by specifically identifying what features and what components are affected and by identifying the type of relationships that govern their configuration. For example, if a large number of relation points appear in the visualisation then the stakeholder immediately comprehends that the decision has a significant impact. Figure 4 shows a conceptual illustration of how such a scenario might be visualised. Using the colour encoding scheme described earlier, we argue that it is easily ascertained that out of the 17 relation points displayed, 8 grey relation points represent features directly related to the decision that would automatically be selected. 2 green relation points represent features that would also automatically be selected due to the selection of other features. 3 red relation points represent features that would be automatically excluded from the configuration. 3 blue relation points represent recommended features that require attention for possible selection and one amber relation point represents a problematic feature. In total, 43 relationships incorporating 3 different models are visualised.

If a large number of red relation points exist then a large number of features are being excluded and perhaps further investigation of the excluded features is advised. Furthermore, if a large number of amber relation points exist then the stakeholder understands that there could possibly be substantial issues associated with this configuration and further investigation is imperative. High risk and high cost features can be treated similarly. Figure 3 shows another conceptual illustration presenting how a high feature exclusion view might be visualised. Here, 6 red relation points represent 6 features excluded due to other included features directly related to the decision.

This visualisation provides a high-level context of all relationships that are impacted by a decision choice. This context can be rotated, zoomed and panned to facilitate exploration of the data and to provide optimum views. This gives the stakeholder an overview perspective allowing them to gauge the current system impact as a whole. By applying distortion techniques, as described earlier, specific
aspects of the visualisation can be highlighted and explored without losing the context. For example, use of blurring and transparency to lowlight all relation points that do not represent feature problematic with feature relationships. This has the effect of highlighting all problematic features where possible issues could arise.

As was discussed in section 4.2, this context can be used as an anchor to drive other views. By clicking on a relation point, alternative views can display more specific information using different visual metaphors.

With the above context, encoding, interaction and exploration techniques in place, the investigation of specific details is afforded. For example, a stakeholder, having highlighted all problematic relationships can click on those relation points and view more specific information such as what feature it is problematic with, providing an issue discovery mechanism.

5. Framework

We are currently developing a tool framework which provides an infrastructure whereby separate models and their relationships can be instantiated and used as the back-end for various visualisations that support a variety of tasks. The details of this framework are out of scope for the current paper but a summary is provided here to relate the approach presented above.

Figure 5 illustrates the outline of the framework. Meta-models describing the SPL data and the relationships that exist allow separate models to be created. The illustration shows three such models, decision, feature and component as well as the existence of the inter-model relationships. These models and their defined relationships represent the back-end data that is used to drive different front-end visualisations that will support product derivation tasks. The approach presented in this paper (“3D Relation View” in Figure 5) will be incorporated as one such visualisation by implementing it using the java3D API. Other possible visualisation examples are shown in Figure 5 such as a simple List View for displaying a list of decisions or a Tree View showing a feature hierarchy and the related component implementations.

The framework uses the Eclipse Modelling Framework (EMF) and a java implementation provides a concrete platform to support the front-end visualisations. It is envisaged that this framework will be developed further to support additional models, relationships and visualisations that will aid product derivation in software product lines.

6. Related Work

Kumbang [13], pure::variants [14], FeaturePlugin [15] and Gears [16] are examples of tools that aim to support product derivation in SPL engineering. All of these approaches primarily employ traditional visualisations to manage the data representation and manipulation of various models and relationships. Foremost of these is the use of simple lists and hierarchical tree views similar to those traditionally

![Figure 5. Framework Outline](image-url)
used for displaying file systems. Such visualisations, though familiar, lack evidence of their effectiveness with large scale product lines. Our approach addresses the tasks of data representation and manipulation from a non-traditional perspective. A relationship-centred visualisation is used in an attempt to manage the complexity and scalability challenges that arise with large software product lines.

The DOPLER [7] tool also supports product derivation and while traditional lists and hierarchical tree views are primarily employed, there exists a mechanism whereby other visualisations can be incorporated. These visualisations can implement graph layout algorithms which again primarily focus on traditional node-link diagrams. While the focus of the tool is on decisions and assets, there is no support for additional models and their inter-relationships.

Unlike any of the tools listed above or others such as XFeature [17], V-Visualize [11] or FeatureMapper [18], our approach attempts to harness the richness afforded by 3D visualisation to address complexity and scalability issues.

Other software visualisation tools such as VISMOOS [19] and MUDRIK [20] provide interesting uses of 3D implementations to support cognition. However these tools and other 3D work such as that by Balzer et al [21] do not support SPL product configuration and are primarily aimed at understanding and not process support.

Other work by Robertson et al [22] and Risden et al [23] are examples of 3D information visualisations where some evaluation of their effectiveness compared with 2D equivalents has been performed. Both papers suggest that in some situations there was no perceived benefit in having a 3D visualisation while in others there was a marked increase in task performance efficiency. This work serves to provide some evidence that 3D techniques can be effective in certain circumstances.

7. Future Work

It is planned to perform further literature review where evaluations of 3D visualisations and/or comparisons of the effectiveness of 3D versus 2D visualisations have been performed.

A specification formalising how this relationship visualisation concept will be implemented is intended. As discussed in this paper, a 3D visualisation environment provides the primary context within which this concept will be exploited. An implementation using the java3D API will be developed and incorporated into the tool framework discussed earlier.

With this visualisation and framework in place, initial user tests are planned to ascertain the effectiveness of this approach to support cognition during product derivation.

8. Conclusion

This paper motivates the need for multiple models in Software Product Line engineering to support management of the large data sets that can exist. It further discusses the requirement and challenges of integrating those separate models to aid different stakeholders in various tasks. A primary challenge is managing the complexity and scale of inter-model (and also intra-model) relationships so that their representation and manipulation is not overwhelming for the stakeholder.

An approach to managing this complexity in relation to product derivation is presented which employs visualisation techniques for the representation and manipulation of the underlying data. This approach focuses on representing the relationships that exist between different models as the primary visual element in the view. A 3D visual environment is employed to render these relationships to support stakeholder cognition during product derivation.

A tool framework that provides an infrastructure within which this visualisation and others can be incorporated is outlined. This framework provides the necessary back end data structures and models to support the visualisation front-end.

It is argued through example scenarios that this visualisation can benefit different stakeholders during product derivation by providing an overview context while allowing specific information to be presented and manipulated without loss of that context.

9. Acknowledgements

This work is partially supported by Science Foundation Ireland under grant number 03/CE2/I303-1.

10. References


