Research Idea

Feature models capture externally visible characteristics of a product. It is proven intuitive and efficient for identifying commonality and variability of elements. The main attraction of any graphical representation is expressiveness. It is very comfortable to comprehend the entire system if it is represented graphically and if it doesn't exceed the usual degree of complexity. However, along with the simplicity and expressiveness another issue should be taken into account; that is unambiguous and complete representation. Every significant detail should be exposed in a model. If these ins and outs were to be represented in only one view and only one model, the system would lose its simplicity and clarity. Therefore another vision is necessary.

On the other hand, use case modelling is an ideal technique for the analysis and specification of functional requirements. Sometimes use case relieves the feature model from 'double duty'. Now use cases gather and describe user requirements under the control of system engineer. A Use Case description will generally include:

1. General comments and notes describing the use case
2. Constraints that must be obeyed. It includes pre-condition, post condition and invariants.
3. Scenarios, meaning the sequential description of the steps to carry out the use case. It may specify exceptional circumstances and alternate paths.
4. Some additional attributes such as version number, stereotypes, status, etc.

Use case model is user-oriented, where feature model is reuser-oriented. Use case models provide 'what' systems a domain do and feature models provide 'which' functionality should be selected while engineering a new system. So one is not an alternative of the other, rather they are complementary of each other. Both of them work together to represent a complete view.

A feature is a distinguishable characteristic of a product and all kinds of features are represented in a feature model. The types of feature can be classified explicitly in a feature model. A feature model represents all functional features as well as non-functional features.

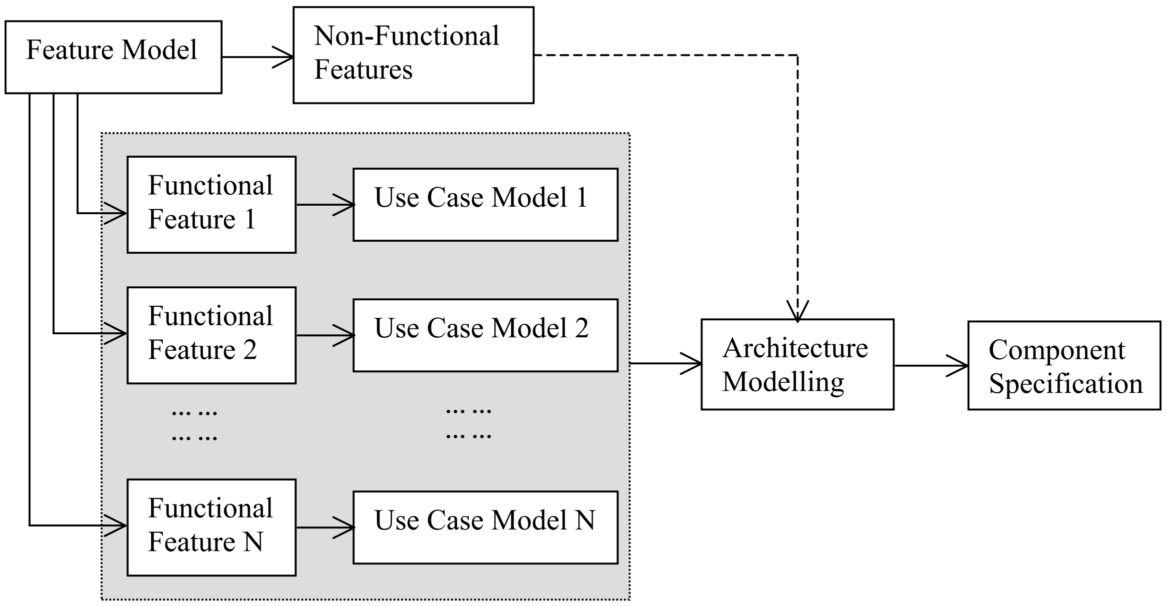

Figure 1: Feature model disintegrated into use cases and integrated into component

It includes common features and variable features. Feature model may specify variation points and variants and their dependency. It can also characterize the major relationship between parent feature and child feature using multiplicity. And many more can be done in a feature model. But what we can't identify in a feature model is the relationship between user and functionality. Fine points of a feature can't be represented in a feature model. Our primary goal is to fulfil these requirements by use case. Apart from the static view of feature model we need a dynamic view like use case model to show some more information about a feature such as, constraints, scenarios, comments and who is responsible for the maintenance of the feature. The internal interactions between the sub features are also necessary to be represented which is possible inside a use case.

Figure 1 describes that one feature model is composed of a number of different features. Only the functional features are taken into consideration to depict a larger interior view with use case. When a group of use cases are always reused together, they can be mapped to a feature and depicted as a use case package. In other words, when a reusable functionality described by the feature is captured by the functionality represented is a group of related use cases. So one non-functional feature produces one or more use cases, consequently one feature model produces several use cases, and we can call this use case package. One important thing to remember is the functionality of the feature must match the functionality of the use case package.

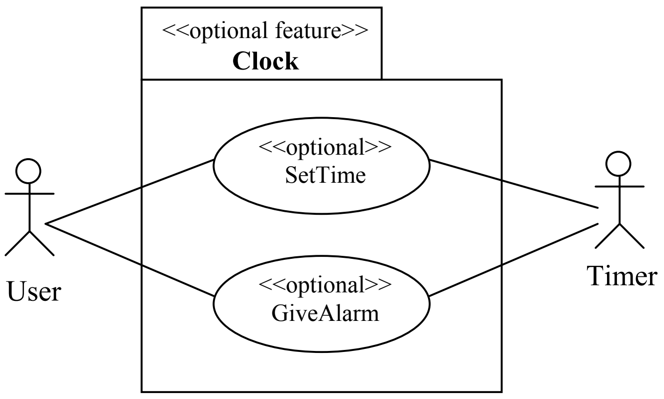

Figure 2: Example of a feature composed of use cases

Consider an example from a cooking stove product line. `Clock` is an optional feature in any feature model. `SetTime` and `GiveAlarm` are two optional use cases that correspond to the `Clock` optional feature.

Case Study

This case study describes the rapid telephone service creation, which is partially adapted from [2]. Figure 3 shows the feature tree view of the telephone system. Variable features are identified by the stereotypes <<variant>>. Relationship between parent feature and child feature are classified by composition, generalization/specialization and variation point and represented by three different symbols. For example, `DialingMode` is a variation point and it is generalized of `Pulse` and `Tone`. For each variation point in figure 3, the association with its child features are denoted by multiplicity. Four types of multiplicity are used here: optional, alternative, multiple optional and multiple alternative. Suppose, the multiplicity associated with `DialingMode` is `1…2`, which means at least 1 and at best 2 features can be selected from this group; i.e. either any one of `Pulse` and `Tone` or both of them can be selected.

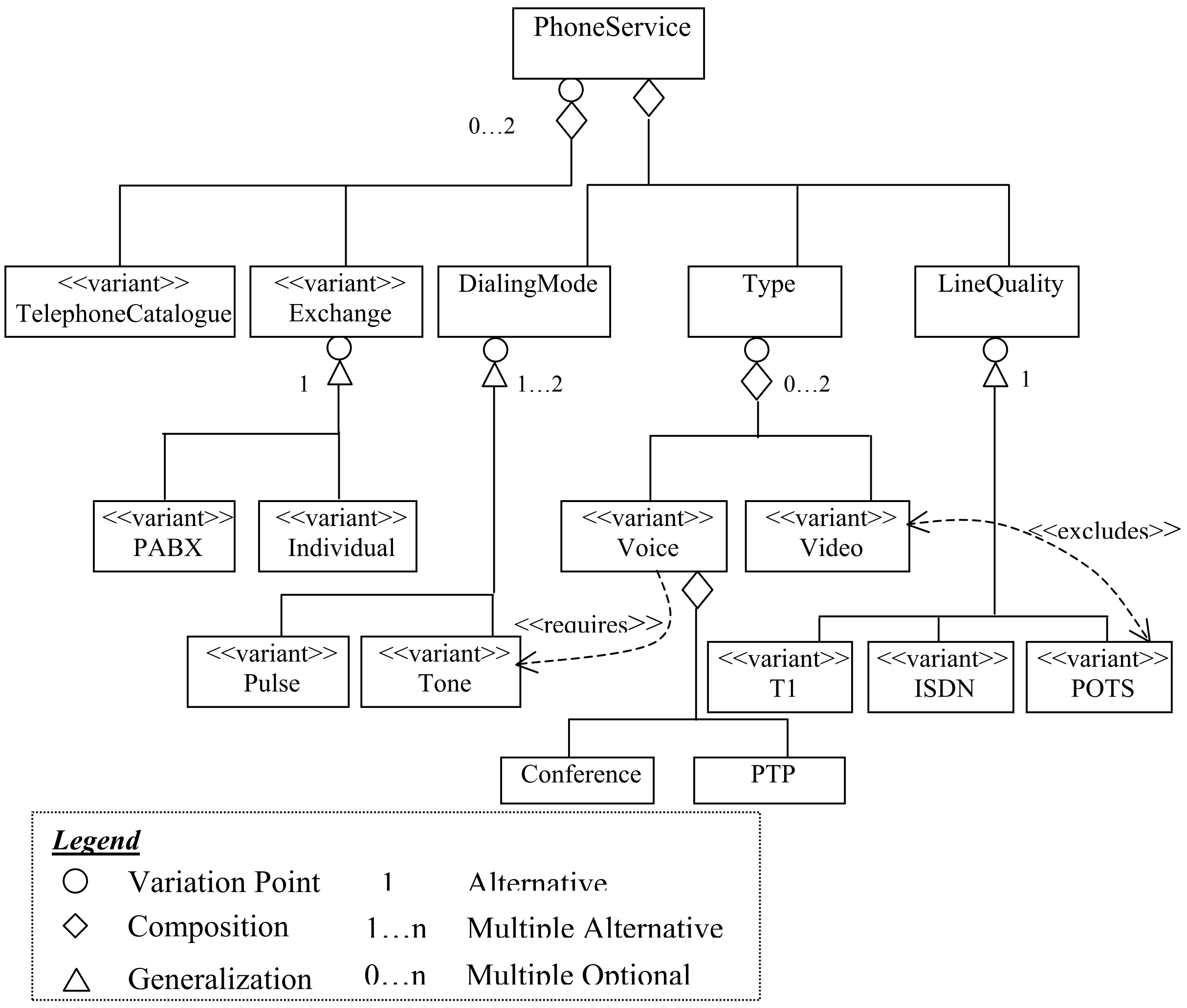

Figure 3: High level feature model: Telephone Service

Major dependency relationships are also represented in this model. *requires* and *excludes* are the most common stereotypes which implies that one feature needs another to be selected and one feature doesn't allow another respectively. In this example, `Voice` requires `Tone` and `Video` excludes `POTS` technology. This case study doesn't deal with complicated dependency representation; instead it is more concerned about showing individual feature's aspect.

Figure 4 describes the internal detail of the optional feature `TelephoneCatalogue`. The major activities that should be done with the telephone catalogue have been characterized with the help of three actors. The order placed by the customer includes three more activities. `RequestCatalogue` is a special case of `PlaceOrder`. Other actors have to do the other jobs like `CheckStatus` and `EstablishCredit`.

Use case is mainly a text document and use case modelling is mainly the ability of writing. The textual representation emphasizes much more than diagram. The sequence, additional path and other information are mentioned in text. Cockburn has developed a template to show detailed functional requirements [3].

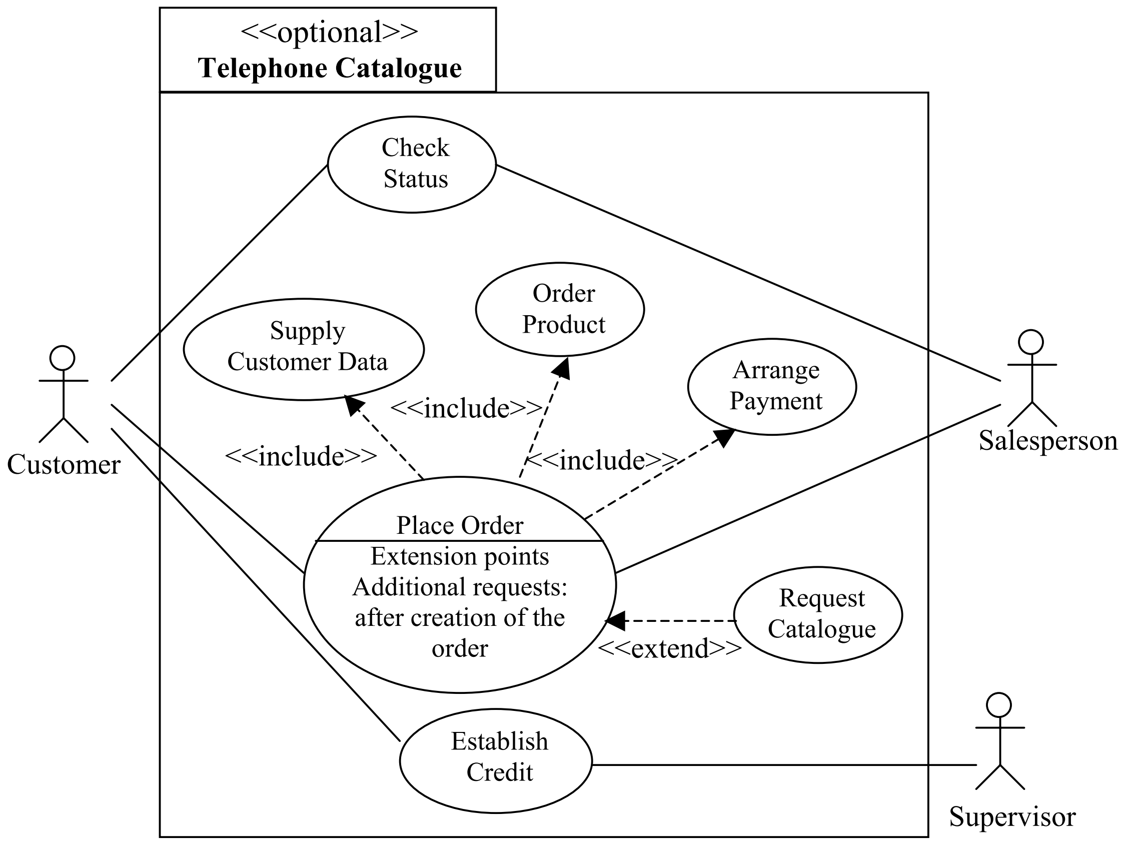

Figure 4: Use case model: Detail view of 'TelephoneCataloque' feature in figure 1.

Our secondary goal is to build a framework for migrating from use case model to product line architecture and then to an individual component. But this is not a straightforward job. There are many more intermediate phases in between use case modelling and specifying component. They are as follows:

1) Use case modelling
2) Business type modelling
3) Component architecture modelling
4) Interface modelling
5) Writing pre and post condition
6) Completing component specification

UML modelling techniques and packages play a significant role in component specification. Use case diagram as well class diagram, business type diagram, sequence diagram, collaboration diagram are required in this purpose. And of course OCL can make it more precise. Use case is mainly used in requirement level. In specification phase class diagram is more useful to specify interface and component.

Reference

1. Brown, Alan W. & Wallnau, Kurt C. Engineering of Component-Based Systems. 7-15. Component-Based Software Engineering: Selected Papers from the Software Engineering Institute. Los Alamitos, CA: IEEE Computer Society Press, 1996
2. Martin L.Griss, John Favaro, Massimo d'Alessandro. Integrating Feature Modeling with the RSEB. International conference of Software Reuse. IEEE Computer Society, Los Alamitos, CA, USA, 1998. pp. 76-85.
3. Alistair Cockburn. Writing Effective Use Cases. Addison Wesley, 2001