Preliminary Results on Effort Estimation using Feature Gap Analysis in Software Product Lines

Sybren Deelstra, Paul Sorenson, H. James Hoover
University of Alberta
Computing Science Department
Edmonton, Alberta, Canada
T6G 2H1
+1 780 492 {3118, 2918, 5290}
{sybren, sorenson, hoover}@cs.ualberta.ca

Jan Bosch
University of Groningen
Department of Computing Science
P.O. Box 800
9700 AV, Groningen, The Netherlands
+31 50 363 3941
jan.bosch@cs.rug.nl

ABSTRACT
Variation points are constructed to accommodate the variety of features within the products of a software product line (SPL). As it's impossible to predict all features up front however, it's most likely there will be a mismatch between some features and the variability provided thru the variation points and variants. We introduce an extendible feature gap model (FGM) as a foundation for determining the capability of a SPL in supporting each product feature set. We explore it's potential as an estimation and evolution tool and present some initial results and experiences.

Keywords
software product lines, variability, variation points, features, feature support.

1 INTRODUCTION

Software product lines (SPLs) are useful for expressing commonalties between related software products. Products within that context tend to vary however and features can be used to describe those differences on a high level [4]. In Bosch' book on SPLs [2] the definition of a feature is specialized for software systems: "[a feature is] *a logical unit of behavior that is specified by a set of functional and quality requirements*." Each feature implements many requirements and a single requirement might cross cut to a number of features, but "*there should at least be an order of magnitude difference between the number of features and the number of requirements for a product line member*". Thus, features are a means to abstract from the requirements in an n-to-n relation.

During architectural design, a software architect is challenged with the task of choosing which features to implement. In an SPL context, it is impossible to predict all of the features one would want to implement in advance of development. To circumvent issues arising from choosing a wrong or incomplete set of features, an architect can delay the design decision to a later stage. A *variation point* [4] is the element of the representation at hand that refers to such a delayed design decision.

Managing variability including all variation points and variants for an SPL is regarded as one of the key success factors for future product development. One way to manage variability is the use of an O-O framework to implement a product line architecture [5]. In this case, we can focus the variability on certain hot spots in the framework and document how to realize this variability using hook points[1]. The problem that still exists even using this well-managed form of SPL development is the determination of which features are well supported (i.e., can be easily implemented) and which features are not supported (i.e., will require substantial additional development effort) in the existing O-O framework.

The aim and contribution of this paper is to introduce a new form of feature gap for SPLs using O-O frameworks as a basis of product development. Our approach is at an early stage of development and experimentation, and therefore this paper is limited to a description of our basic feature gap model (FGM) and how we plan to enhance and evolve this model. Our goal for the FGM is to provide an accurate effort estimator for work required to develop a product, based on its identified features, using an O-O framework as an expression of our SPL architecture. In this sense, we are proposing a light-weight feature selection technique that is combined with an effort prediction model. As a selection technique, it is simpler and more focused on product line development than SQFD (Software Quality Function Deployment [8]). Our effort model is much simpler than

[1] In other words, hook points are documented variation points.

COCOMO [6], because it is calibrated relative to each specific product-line framework.

The remainder of this paper is organised as follows. Section 2 provides a short introduction to the Prothos framework, which is used as the basis for SPL described in this paper. Section 3 introduces feature support concepts and section 4 describes the feature gap diagram and its use in FGM. Sections 5, 6 and 7 describe how the FGM can be applied to software product line selection, initial feature selection and product line evolution. Section 8 describes our use of a feature support questionnaire and section 9 presents some of our initial findings based on the questionnaire results. Finally, section 10 discusses our plans to validate the FGM (Feature Gap Model).

2 THE PROTHOS FRAMEWORK

Prothos is an OO framework and part of a product line architecture for web-delivered database-centric applications and is used to build commercial production applications.

Prothos provides a minimalist environment for implementing a web-based client/server RDBMS application that wraps order processing business classes into a working product. It supports the selection, display, editing, and posting of persistent business data that has been structured into a set of instances of business classes. The business classes implement the application's business rules, which are responsible for the application-specific processing.

The User Interface Manager is a subframework of Prothos that coordinates the user interface to the set of persistent business classes. These business classes encapsulate access to the business services and obtain their persistence by inheriting from a second Prothos subframework, the Persistent Object Manager.

3 DEFINING FEATURE SUPPORT

During architectural design, variation points are introduced in the product line to accommodate the flexibility regarding changes in future products and product versions. In Prothos, variation points are realized as hooks in the framework that indicate how and where variants can be added in order to develop a new product. Because the Prothos architects could not possibly predict all future product requirements and features in advance, there is inevitably a mismatch between the variability that is well supported in the framework and the variability needed to support the requirements and features of the new product.

Example: A good example of a product requirement that is well supported in the existing Prothos framework is the creation of a new business class and the capture of instances of that class (class objects) during execution. However, if the workflow around that new business class is rather unique and requires some special processing not previously supported in the framework, then the mismatch between existing framework support and what is required by the application is large

This variability mismatch translates into feature gaps: a mismatch between a feature and the support provided by the product line architecture. To reflect a measure of support for a particular feature in the product line architecture we have chosen a five value scale ranging from 'full' to 'good' to 'basic' to 'weak' to 'no' support. Unfortunately, there is no common understanding what defines the amount of support. Froehlich's Hooks Model [7] identifies three levels of support for hooks, namely option, template and open hooks; however we are only just beginning to determine how we can incorporate these notions in our FGM and therefore will not be reporting on that work in this paper. Our report in this paper will explore the potential of FGM as an estimation tool by applying it in an after-the-fact study and not as part of an active process in SPL development. Specifically, we examine how applicable FGM could be in aiding effort estimation and product development through an observational study of two products developed using the common Prothos framework as a basis for product line development.

4 FEATURE GAP DIAGRAM

An important part of using the FGM is the ability to visualize the gap between a framework and a product feature set. As a consequence, we have plotted the features against the support classification described above to form a *feature gap diagram (FGD)*. Examples of FGDs for two example product developments that are currently using Prothos are shown in Figures 1a and 1b. The two products, which are being implemented by two student teams in our CMPUT 402 course, are Product Team 1, a Strategic Initiatives Management System (Cafe SIMS), and Product Team 2, a Conference and Workshop Management System (Cafe CWMS)

(Note: Product Team 1 choose to record their support assessment in just three of five categories, because they felt this was as accurate as they could be in their initial estimates.)

An FGD depicts the relative importance of a feature against the perceived support for that feature in the existing framework. For example, in Figure 1b, feature 2 is viewed as a ***key*** feature that is ***fully*** supported in the framework; whereas, feature 4 is a ***desired*** feature that has ***no*** support in the framework. Currently, we are investigating the full benefits of using this form of gap visualization for important aspects such as *SPL selection, initial feature selection,* and *SPL evolution*. These aspects are discussed in the following three sections.

5 SOFTWARE PRODUCT LINE SELECTION

Perhaps the most obvious way to use feature gap diagrams is in early meetings to assist customers in visualizing the suitability of an SPL architecture with regard to the features identified in their requirement set. This variability management technique is relatively easy to implement without spending significant time in detailed effort prediction. In the theoretical worst case (in which the SPL architecture does not support a proposed product) all or most of the features would fall on the right-most section of the FGD, while in the optimal case (near perfect SPL architecture match) all features would fall in the left-most sections.

We can observe from the feature gap diagrams for both student products (Figures 1a,b), that the Prothos framework is reasonably well suited for Product 2 and less well suited for Product 1. Indeed, further detailed analysis is warranted before making a decision to use Prothos for Product 1. The initial feature selection process described below could certainly be of some help in this regard.

6 INITIAL FEATURE SELECTION

It is typical in the early stages of feature identification for a new product to come up with many more features than one could possibly implement in the initial product rollout. Choices of which features to include in the initial product depend on several factors that are both customer driven and technical in nature. Our hypothesis is that fully supported features *generally* require less effort to implement than features that are not well supported. Feature gap diagrams can be used for initial (coarse grained) feature selection by selecting and discarding sections as depicted in Figure 2a. In this case, three categories (fancy/weak, fancy/no, desired/no) are eliminated and two categories (fancy/basic, desired/weak) should be considered as candidates for elimination.

The nature of this selection and discarding of features undoubtedly depends on the market and/or the culture in the developing organisation. In a mobile phone market, for example, the emphasis is on feature-rich products and therefore the objective is to discard very few features (see Figure 2b). On the other hand, small organisations (like the student team product development shown above or computer game products) would discard features rapidly in order to get the product released at a precise time (end of term or in time for the Christmas market).

After this coarse grained selection, a more fine grained 'per feature' effort prediction is often required to add or discard additional features from the final product feature set, for at least two reasons. First, choices in the mid-section of the diagram (e.g., the sections surrounded in a bold line in Figure 2a) can be difficult. Second, a selected feature can still consume a large portion of the total effort as the feature gap diagram does not tell us anything about the distribution of effort within each section or among the different kinds of features.

To address these shortcomings we recommend that an initial FGD be revised in the following way. First, framework users are asked to provide an initial estimate of the amount of total effort (in person days) to implement each feature in the selected feature set, although in reality, the feature might be implemented in phases. Features that are not selected are not assigned an effort estimate (i.e., their initial estimated defaults to *undefined*). Each dot representing a particular feature on the FGD is then sized to reflect the relative amount of effort to implement that feature. Feature dots would not appear for features that are not selected. As an example, Figure 3 depicts the relative weights for features selected to be in the initial version of Product 1. By using these initial effort estimates, choices of which features to include/exclude in a next release can be decided based on total effort calculations as provided in the enhanced FGD.

7 PRODUCT LINE EVOLUTION

From time to time we will need to evolve our framework based on our experiences in trying to deliver several products from the framework. One way to do this is to combine FGDs for all products, either with existing feature sets or sets of predicted feature, into a single (total) FGD and use the resulting diagram to assist in determining an evolutionary path for the underlying framework that supports the SPL.

We are currently investigating how to represent the timeline and the decision making process. Of particular concern is that fundamental (cold spots) on the framework may change. As the framework evolves, the subframeworks can be refactured and thus the implementation of a feature can change. Features can appear and disappear. This would have ramifications on all existing products that have been built from an old version of the framework

An interesting and promising approach for evolving feature support in the framework, is to group features based on the commonality they have with respect to the hooks that would be enacted to implement the features. We plan to investigate this approach by applying this analysis across several feature sets that have been implemented for Prothos-based products.

8 FEATURE SUPPORT QUESTIONNAIRE

To begin to measure the effectiveness of the FGM approach, we constructed a questionnaire in which we ask the framework users to identify for each feature in their original feature set, the category and the amount of support for that feature in the product line. In the questionnaire, we asked the framework users to classify the features into the three FGD categories that relate to the perceived need for the feature, namely, "key", "desired" or "fancy" feature. Key features are the core features that identify the application, while fancy features are non-essential, low

priority features.

The questionnaire also contained the definitions for the feature categories and the following guideline to the 'amount of support' classification: "A lower amount of documentation of and/or a less perfect fit with the corresponding variation point results in a lower amount of support". The term 'fit' is not further defined; however to assist in understanding their notion of fit, each subject was urged to explain their classification of each feature in more detail.

9 INITIAL RESULTS

We sent by email the questionnaire outlined above to the two student teams involved in developing separate applications using the Prothos framework. The results were sent back with email as well. The teams identified a total of 38 and 26 features[2] in their feature set, respectively. The questionnaire resulted in the class distribution below. Note that the number of features and, where appropriate, brief explanations of the classifications provided by the product teams are included.

Full support (10+10)

- Features already implemented in the framework (18)
- Variants available, but requires small modifications (1)

Good support (0+2)

- Existing variants require minor modification (2)

Basic support(15+4)

- No available documentation on variation point or variants (5)
- Variants requires modification, but no documentation available (4)
- Variants not portable and no documentation available (3)
- Modification of existing variants required (2)
- Adding of functionality to existing variants required (2)

Weak support (0+8)

- Significant amount of modification is required (3)

No support (13+2)

- No variation point exists (8)
- No variation point found (5)

Both student teams identified the amount of documentation as the major factor in classifying the features. Even if the variants are available, the feature is classified into the 'basic' class. No documentation however, does not imply that the corresponding variation point does not exist, which is reflected by the 'no variation point found' remark for the 'no support' classification. Proper variability management is the key here and is an area that requires more research, e.g. on how to describe and represent the variation points on all abstraction levels.

10 VALIDATION

As an initial step towards validating our FGM approach, we mapped the set of features chosen by the product teams using a similar SQFD feature selection strategy onto the FGDs shown in Figures 5a and 5b. Product Team 1's initial feature set is circled. Product Team 2 is proposing three cycles of product development. The features to be implemented in the first cycle are circled. The second cycle features are encased in squares. An initial analysis of these results suggests some alignment exists between the FGM approach and SQFD analysis used by the product teams.

As soon as both teams finish the applications we will give them a similar questionnaire to see how they experienced the amount of support and if their opinions regarding the factors have changed. We'll also ask framework experts to fill out the same questionnaire for both application feature sets and have a discussion with industry experts regarding their point of view on the amount of support for features.

The questionnaire results to date suggest that having good documentation is very important to assessing perceived support. We believe other factors such as variant binding time or type of feature should play an important role as well. However, none of the team members is a framework expert, which might explain why the teams value documentation over other factors. We expect the outcome of this research will point to several other factors that influence feature selection and effort estimation, all of which will allow us to improve our FGM approach.

REFERENCES

1. L. Bass, P. Clements and R. Kazman, *Software Architecture in Practice* (Addison-Wesley), ISBN 0-201-19930-0, May 1999
2. J. Bosch, *Design & Use of Software Architectures, Adopting and evolving a product-line approach* (Addison-Wesley), ISBN 0-201-67494-7
3. J. Bosch, G. Florijn, D. Greefhorst, J. Kuusela, H. Obbink, K. Pohl. "*Variability Issues in Software Product Lines*", Proceedings of te Fourth Workshop on Product Family Engineering, October 2001, to be published at Springer LNCS
4. J. van Gurp, J. Bosch, M. Svahnberg, "*On the notion of Variability in Software Product Lines*", Proceedings of

[2] E.g. *full history logs*, *a common document repository*, *list creation*, *conference registration*

The Working IEEE/IFIP Conference on Software Architecture (WICSA 2001), pp. 45-55, August 2001.

5. H.J. Hoover, T. Olekshy, G. Froehlich. and P.G. Sorenson, *"Developing Engineered Product Support Applications",* Proceedings of the 1st Software Product Line Conference, sponsored by the Software Engineering Institute, Denver, CO, Aug. 2000, pp. 451-476. Published as Software Product Lines - Experience and Research Directions}, P. Donahoe, ed., Kluwer Academic Publishers, 2000.

6. Boehm, B. Software Engineering Economics, Prentice Hall, 1981.

7. Froehlich, G., Hoover, H.J., Liu L. and Sorenson, P.G. *"Hooking into Object-Oriented Application Frameworks"*, Proc. 19th Int'l Conf. on Software Engineering, Boston, May 1997, pp. 491-501.

8. Haag, S., Raja, M.K., and Schkade, L.L. *"Quality Function Deployment Usage in Software Development,"* CACM, Jan. 1996, vol.39, no.1, pp. 42-49.

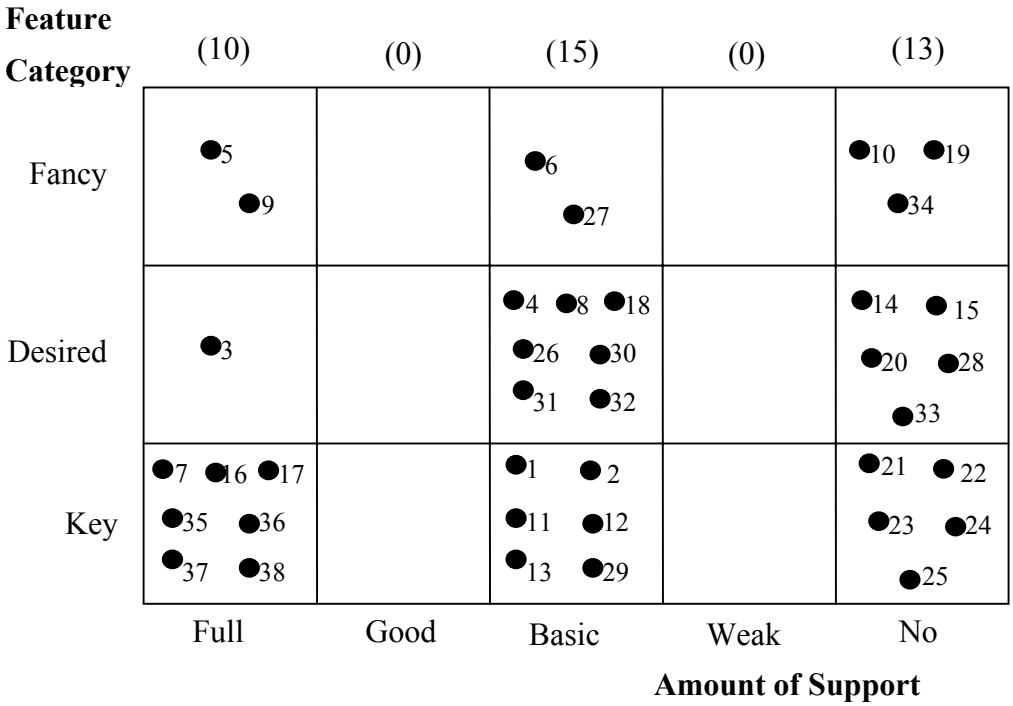

Figure 1a: Feature Gap Diagram for Product 1

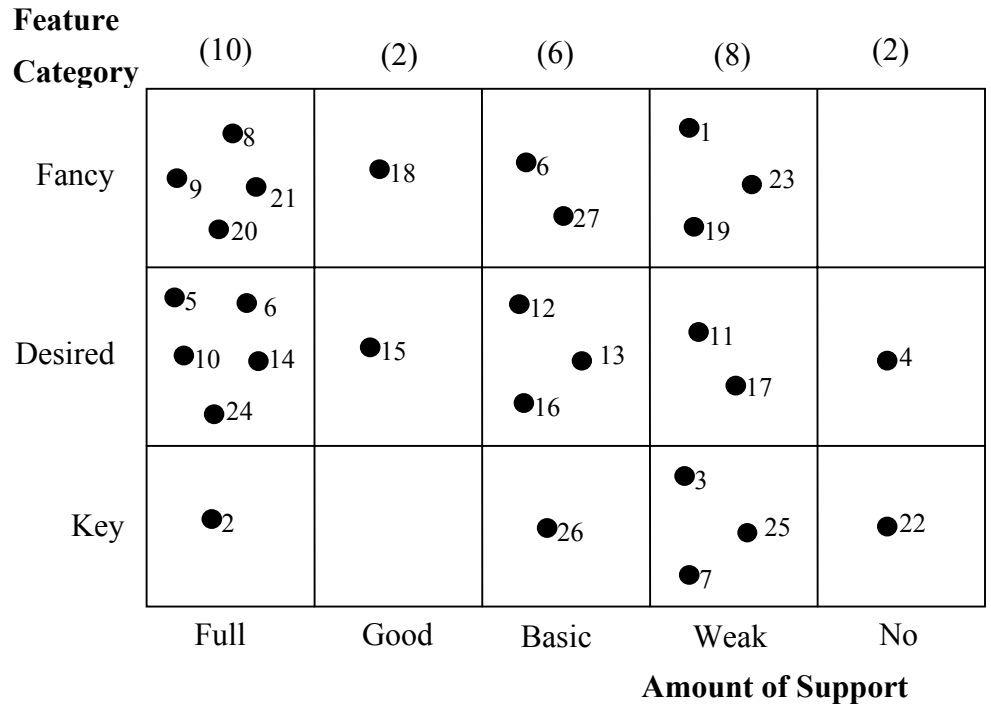

Figure 1b: Feature Gap Diagram for Product 2

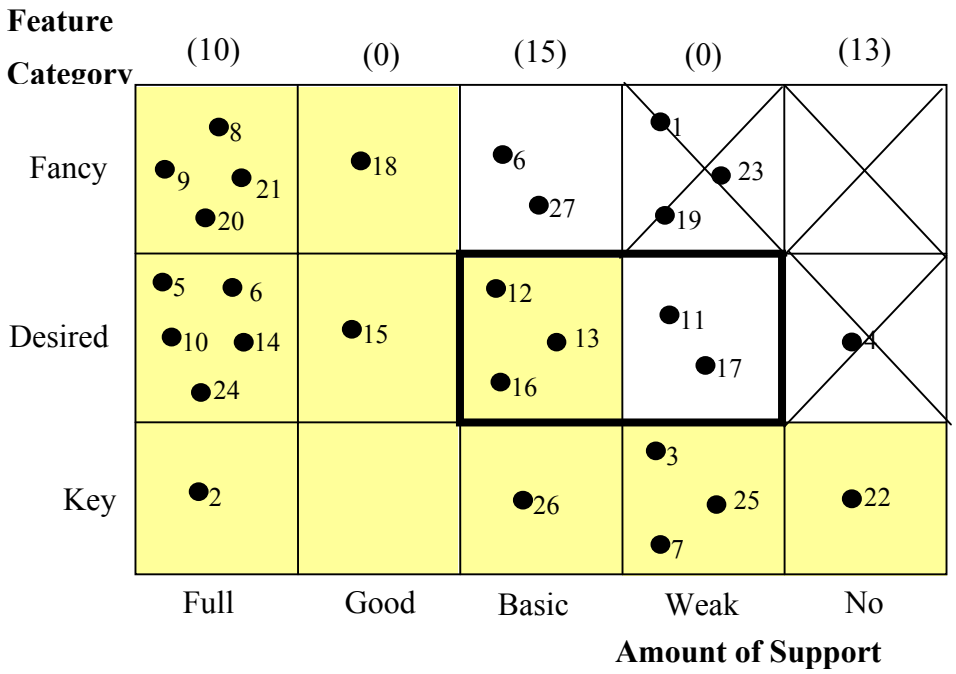

Figure 2a: Feature Gap Discard Diagram for Product 2 illustrating aggressive discard strategy legend!

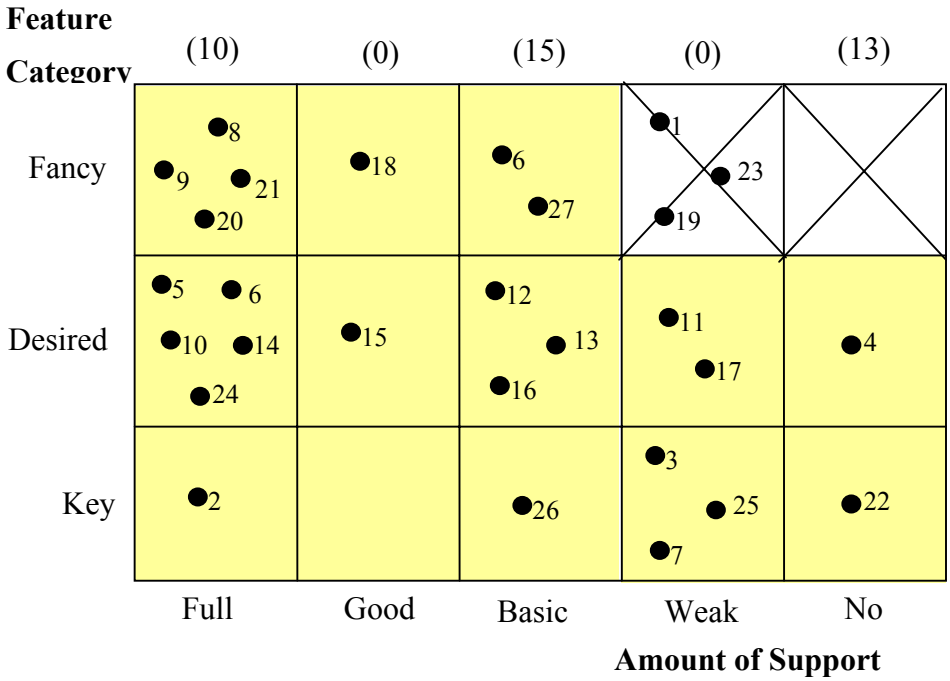

Figure 2b: Feature Gap Discard Diagram for Product 2 illustrating conservative discard strategy

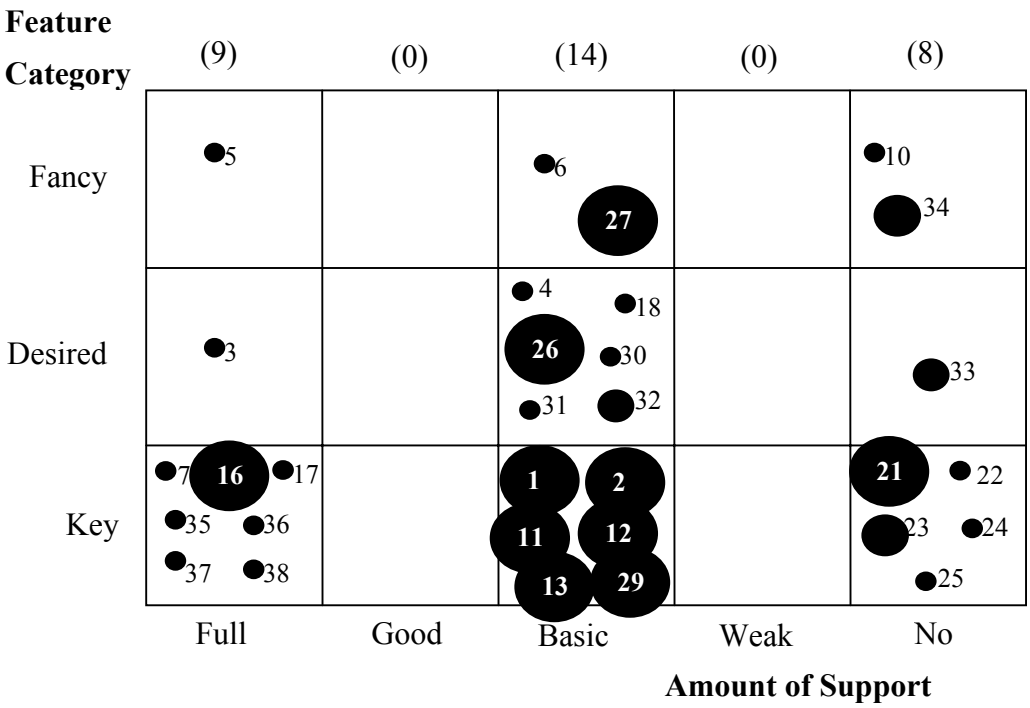

Figure 3: Feature Gap Effort Diagram for Product 1 depicting relative weights of selected features

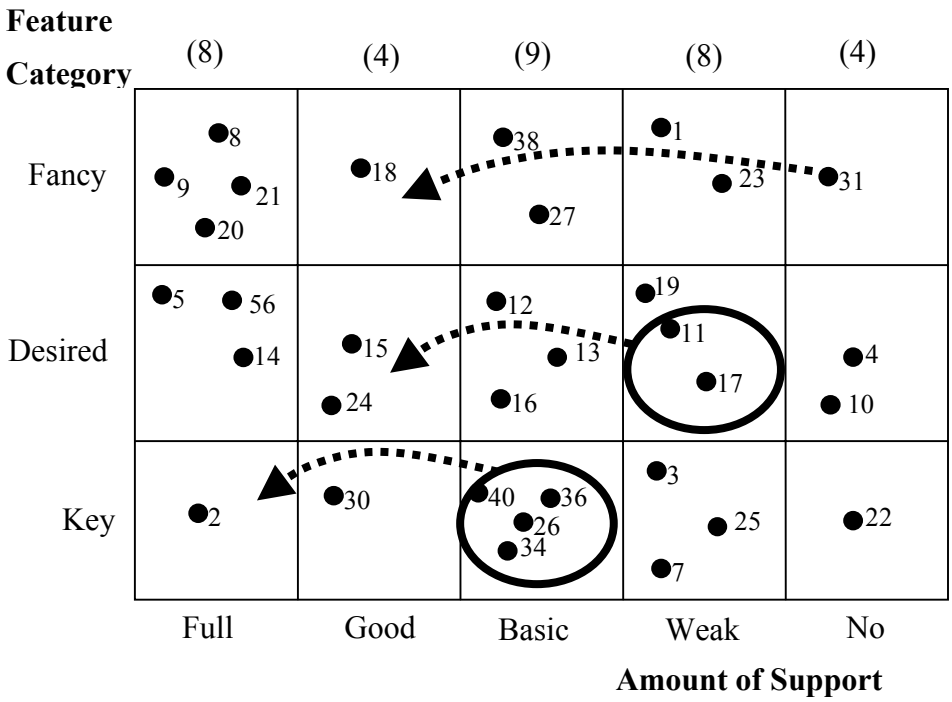

Figure 4: Feature Gap Evolution Diagram in support of Product Line Evolution

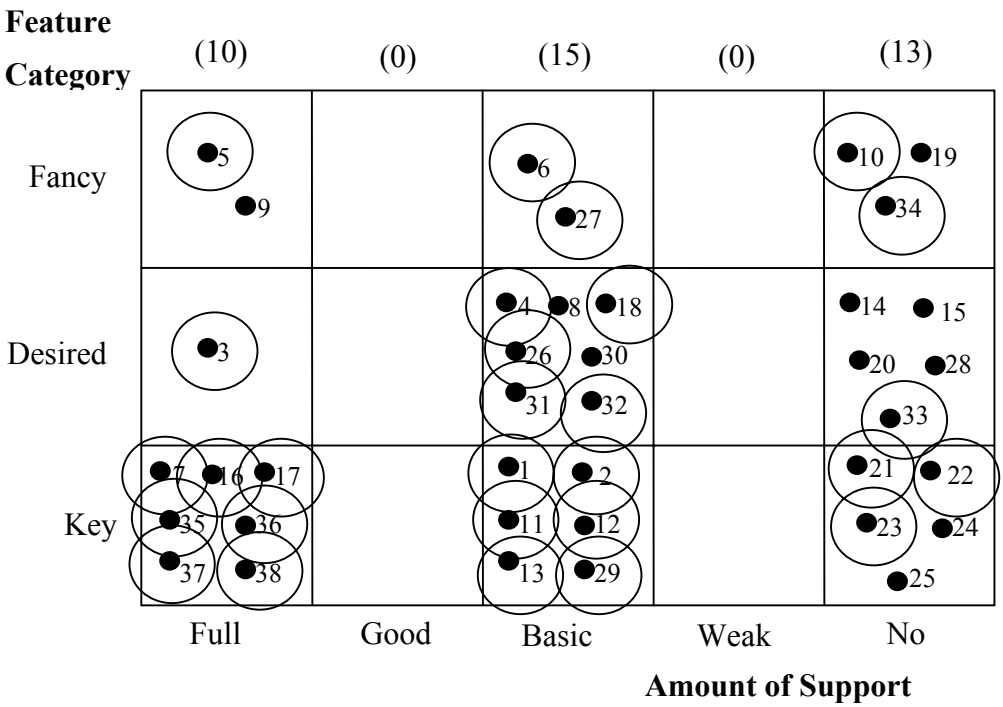

Figure 5a: Initial Feature Set Selection for Product 1

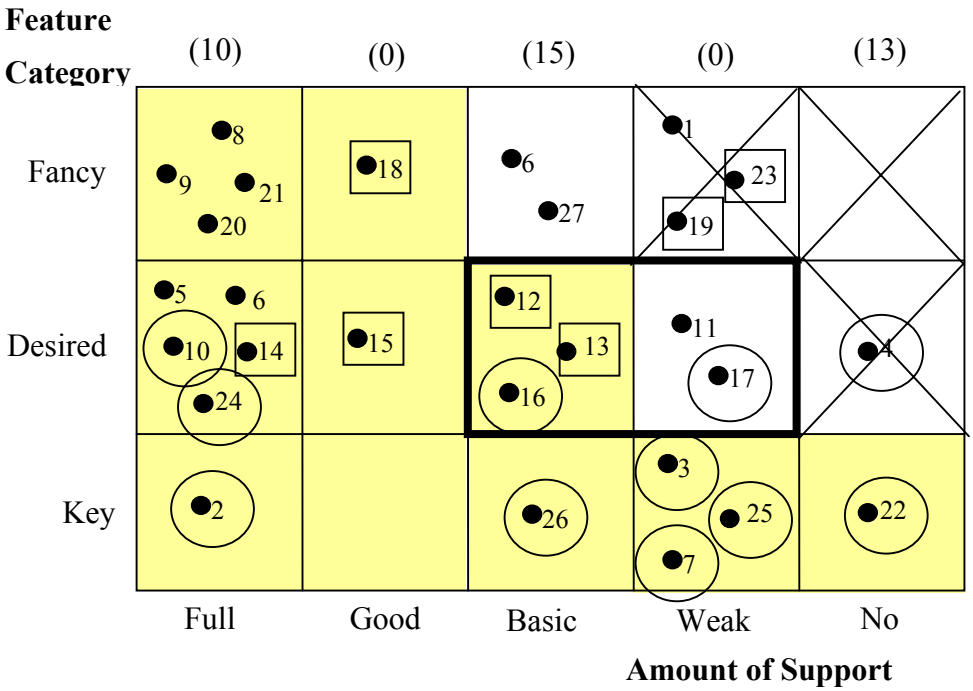

Figure 5b: First and Second Cycle Feature Selection for Product 2