Background

**Software product lines(SPL):**

A software product line (also sometimes called software product family) is “a set of software intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets [artifacts] in a prescribed way” [1].

A family of software products having some commonalities and variabilities is termed as software product line family. Commonality is a property shared by all products of the family [2], whereas, variability points out the differences among the products that belong to a software family [3].

[1] <http://splc.net/fame.html>

[2] <http://www.sei.cmu.edu/productlines/>

[3] FoSE\_SPL\_sota\_classification

**Software Product Line Engineering(SPLE):**

SPLE empowers organizations and businesses to implement variability in their software systems to generate a family of software products at lower cost, in shorter time and to improve productivity and quality when compared with the development of single systems.

Software product-line engineering constitutes mainly two engineering phases: domain engineering and application engineering as shown in figure Number.

Domain Engineering:

Domain engineering consists two tasks: domain analysis and domain implementation.

Domain Analysis: In this task, a variability model called as feature model is constructed by considering both the commonalities and the differences (variability) of a set of software products.

A feature model is represented as hierarchical data with tree structure, called as feature diagram[38]. “Each feature is a characteristic or end-user-visible behavior of a software system” [4]. As an example, consider the smart-home feature diagram in Figure~\ref{fig:efm} where boxes denote features, and links illustrate the interdependencies between them. There are common features found in all products of the PL, known as \textit{mandatory} features, such as \texttt{illumination}, and variable features that allow the distinction among products in the PL, referred to as \textit{optional} and \textit{alternative} features, such as \texttt{security} and the group \texttt{sensor}, respectively.

In addition, FMs often contain \textit{cross-tree constraints} (CTCs).

CTCs add new feature interdependencies to the feature diagram, by restricting feature combinations (\textit{e.g}., \texttt{sensor $\rightarrow$ alarm}). Furthermore, dashed boxes extend FMs with extra information about features (\textit{i.e.}, NFPs).

This type of models where NFPs are included is called \textit{Extended Feature Model} (EFM)~\cite{BSR+:10}.\\[-5pt]

[4]ABKS13. Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. Feature-Oriented Software

Product Lines - Concepts and Implementation. Springer, October 2013.

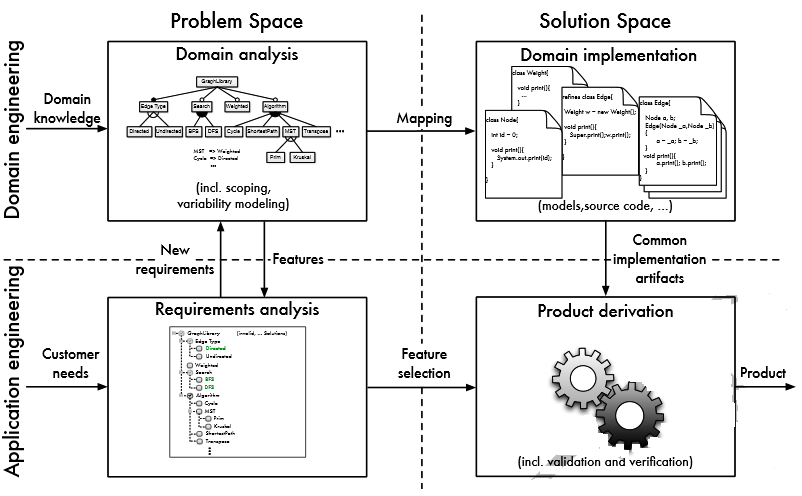
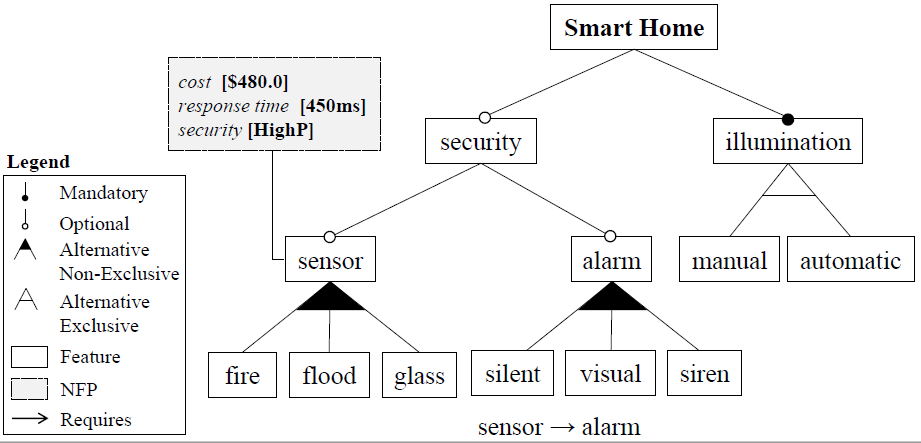


Fig. An overview on software product-line engineering [4]



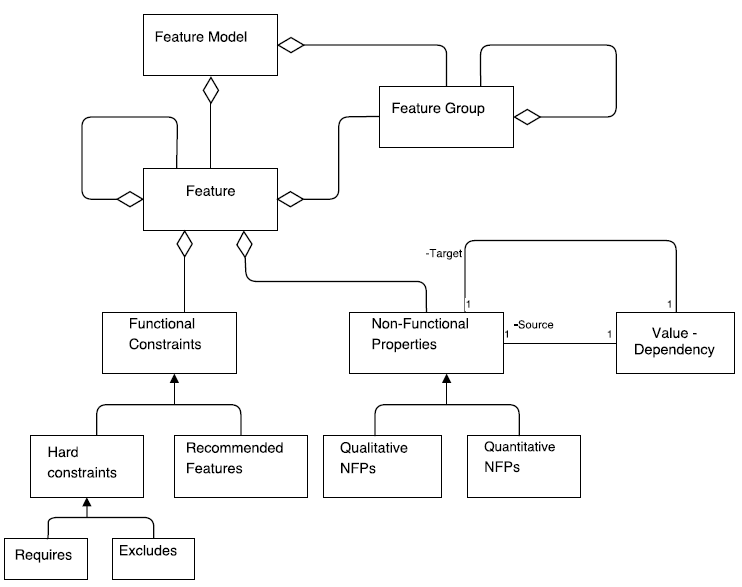
“A feature in extended feature models is a logical unit and has both functional and nonfunctional properties” [1]. UML diagram of an EFM is shown in figure NUMBER is adopted from [toward], where feature in the feature model has both functional properties

and NFPs. Functional properties are divided as constraints between the features, Hard constraints and recommended features, these are features recommended with 5 star view to assist the user in configuration process [feature recommender system].

Non-functional properties(NFP's) can be divided into two Categories: Qualitative and Quantitative[toward]. A quantitative NFP is measurable and can be represented as a numeric value [36]. Memory usage, Cost, and Performance are examples of quantitative properties. A qualitative NFP is a second category of NFP, which cannot be exactly measured [6]. These qualitative NFPs are represented as one of the six qualifier tags: High negative, Medium negative, Low negative, Low positive, Medium positive, and High positive. Security, customer satisfaction and international sale are the examples of the qualitative non- functional properties [toward].

1. J. Bosch, Design and Use of Software Architectures – Adopting and Evolving a Product-Line Approach,

Addison-Wesley, 2000.



Domain implementation:

Code for each feature in the EFM can be implemented using different diverse programming languages to make each feature as a re-usable software asset.

**Application engineering:**

Application engineering comprises two tasks:Requirements analysis and product derivation.

Configuration problem(toward):

**SPL Tools:**

In the SPL research community, there are several SPL tools are proposed to assist domain experts, application engineers, and stakeholders in the phases of the SPL life cycle. For example , *FeatureIDE* [Thüm et al. 2014] , *Feature Model Plugin (fmp)* [Czarnecki & Kim 2005], *Feature Modeling tool* [Fernández et al. 2009], *SPLOT* [M.Mendonca et al. 2009], *Fama Tool suit* [D.Benavides et al. 2007] , and *Pure:variants* [O.Spinczyk et al. 2004].

Mechanisms such as visualization and interaction techniques as well as feature-based recommender system play an important role in reducing the burden on the user to achieve his target product in the configuration process. The support of SPL tools in realizing these mechanisms are discussed below in detail.

**Visualization:**

A picture or an image or visualization plays an important role in human perception and cognition, which also includes awareness, reasoning, and learning. There is a saying that “a picture is a worth of thousand words”, which is emphasized by many philosophers and scientists throughout the centuries

“...thought is impossible without an image “

(Aristotle, 350 BC)

“Imagination or visualization, and in particular the use of diagrams, has a crucial part to play in scientific research “

(Ren´ e Descartes, 1637)

“The drawing shows me at one glance what might be spread over ten pages in a book”

(Ivan S. Turgenev's novel Fathers and Sons, 1862)

Representing data in visual form can help people to better explore, analyze, and understand it, thus transforming the data into information. We can convey complex information in an intuitive way through visualization. Visualizations can be represented as text as well as two- or three- dimensional representations. Gershon [Ger94] defines visualization as follows:

“Visualization is more than a method of computing. Visualization is the process of transforming information into a visual form, enabling users to observe the information. The resulting visual display enables the scientist or engineer to perceive visually features which are hidden in the data but nevertheless are needed for data exploration and analysis.”

From both the literature in visualization community and the observations during this project, there are various aspects must be taken into consideration while creating a visualization:

1. Convey meaningfully and required information

2. Use of color in visualization: Color mapping is a very important visualization technique, which must not only make the visualization, visually attractive but also depict the desired information in a clear way **[using color in vis].**

3. Perception and cognition: Light is a kind of electromagnetic radiation. Color is the human perception of light. The human eye is perceptive to some colors, design patterns, motion and graphical representations as presented by **Diehl[].**

4**.** Choosing an appropriate visualization paradigm: Hernandez et.al [what for: classification of visual pardmgs] proposed a two-dimensional classification of visual paradigms as in figure Number based on analysis of internal relationships among data and the ideal visualization paradigm to fit the specific use case and to meet the user needs. One dimension is based on the nature of data ( Trees/Hierarchical data and Networks/No Hierarchical data) and the other dimension is based on finding out which of the features (location, attributes, relationships and time) is important for the specific use case.

5. Focus on more relevant parts, and provide attention to detail and do not load with large amounts of data.

In order to make sure that the visualization illustrates the information what we want and to help the users to arise to a decision in solving the problem.

**Information Visualization:**

In the current modern world, computers and internet have become part of our life and computers have become an essential tool for creating visualizations and helping the user to better understand complex phenomena.

According to S.K.Card et al. [Readings in information visu 1999],

“Information visualization is the use of computer-supported interactive visual representations of abstract data to amplify cognition. Its purpose is not the pictures themselves, but insight (or rapid information assimilation or monitoring large amounts of data).”

These computer-supported interactive visual representations have promise for the reasons as stated by S.K.Card []: 1. It brings increased resources to the human in the form of perceptual processing and expanded working memory. 2. It can reduce the search for information. 3. It can enhance the recognition patterns. 4. It enables the use of perceptual inference and perceptual monitoring.

More information on visualization in general and on information visualization, in particular, can be found in various related books [CMS99, War00, Spe01, Che04].

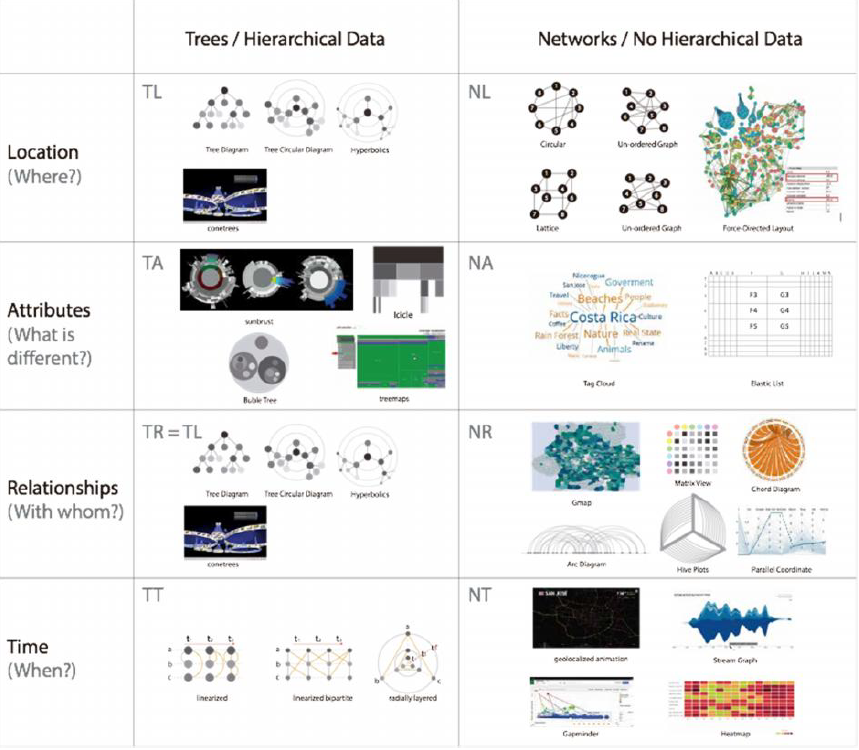


Fig. Classification of Data Relationships for Visualizations [ ref]

**Software Visualization:**

**Visualization Pipeline:** This Pipeline presents various phases of the visualization process in a software tool. Firstly, the data acquired from the various sources like source code, its design, user documentation, state changes during its execution, test results. Secondly, the data collected is analyzed using various kinds of analysis, such as filtering, static program analysis, or statistical methods, can be used to reduce the amount of data and to focus on the important parts. Lastly, the resulting data is mapped onto a visualization paradigm to render image(s) onto the screen [Stephan Diehl]. The visual steering provides the ability to control the first and second steps of the pipeline based on the graphical output produced earlier [JPH+99, MvWvL99].

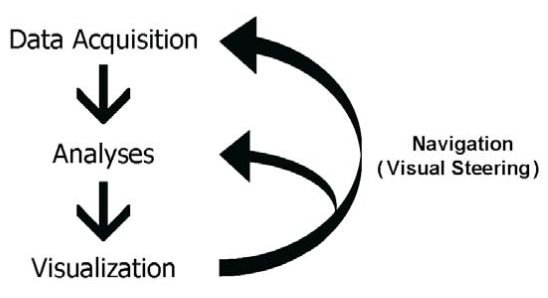


Fig. The visualization pipeline []

Bassil and Keller [BK01] found the following reasons why practitioners apply software visualization, from the survey conducted with 107 participants mostly from the industry

* savings in time and money;
* better comprehension of software;
* increase in productivity and quality;
* management of complexity;
* to find errors.

**Visualization in SPL tools:**

In the SPL research community, a wide variety 0f tools are developed with visualization and interaction techniques for assisting domain experts, application engineers and stakeholders in different tasks of SPL life cycle [ Cawley et al. 2009; Siegmund et al. 2014; Nestor et al. 2007 ].

In addition, there are SPL tools which support the visualization of properties of a feature(functional or NFPs or both) and the visualization and interaction mechanisms to alleviate the feature model configuration challenges.

Thüm et al. [2014] presented *FeatureIDE*, an open-source extensible framework for feature-oriented software development (FOSD) based on Eclipse. Pereira et al. [2016a] proposed improvements to alleviate the configuration problem by extending the tool by Thüm et al., through visualization mechanisms, such as *information hiding* and *decision propagation*. Information hiding mechanisms facilitate the users to focus on their relevant configuration space. They provide a focused view, in which, only upon a selection of a feature, its' sub-features are shown by the expand algorithm.

Martinez J et al. [2014] presented an interactive visualization paradigm, called *FRoGs* (*Feature Relations Graph*) to represent feature constraints in SPL. FRoGs is also built on the top of FeatureIDE [ T.Thüm et al. 2014 ]. ], It shows how a specific feature is in connection with the rest of the features in terms of constraints and different stakeholder perspectives(i.e., customer, environment, functions, design, behavior, and components).

Nestor D et al. [2008] introduced integrated meta-model and employed visualization techniques to address imperative SPL tasks such as variability management and product derivation. The visualization techniques in this approach are presented in the section Related work.

Asadi M et al. [2014] proposed the feature models extended with the notion of NFPs and their\textit{ Vis-fmp tool } is an extension of the \textit{fmp tool} \cite{Czarnecki2005Cardinality-basedReport} , which supports several visualizations and interaction techniques like overview, zooming, filtering and highlight, detail on demand, interactive configuration, are described in related work section.

Several other SPL tools which support the visualization and interaction techniques are presented in the related work section in more detail.

**Effects of visualization and interaction techniques :**

Asadi et al. 2016 empirical study revealed that by applying visualization and interaction techniques in the SPL tools significantly decreases the time required for the configuration tasks of simple and complex feature models. The results also showed that the ease-of-use and ease-of-learning are better for the visually-enhanced tool implementation.

**Visualization Techniques for Application in**

**Interactive Product Configuration**

Pleuss et al. [] investigation on SPL tools in the context of Visualization Techniques for Application in Interactive Product Configuration revealed that the visual techniques like clustering, decision trees, treemaps, cone trees, tables, ow maps, and UML diagrams are mostly applied in the configuration process. In addition, they argued that the complexity is a limiting factor in the successful adoption of PLE and urged the need for better techniques for handling large and complex product line models (i.e., in product configuration) are required.

Moreover, the available tools in the literature still lack in the proper visualization and interaction techniques, which are essential to address the \textit{cognitive complexity} that is implicitly present in the industrial feature models with hundreds or thousands of features in a more intuitive way.

**Recommender Systems:**

In today’s world, the increase in the use of computers and world wide web/Internet in conjunction with the information overload problem has served as a driving force for the development of recommender systems technology. Recommender Systems are playing a crucial role in providing required information to the users by filtering out large amounts of data through different recommendation algorithms. The most common classes of such algorithms are content-based recommender systems and collaborative filtering algorithms.

A user is recommended with the items, similar to the content of the items that he was searched before by using his user profile. this approach is implemented by content-based algorithms. collaborative filtering algorithms recommend items to a user based on the items recommended by a group of other users when their preferences are matched.

Collaborative filtering algorithms find out the items recommended by a group of other users, recommends to a user, given the preferences of the user are matched with the group of users.

Czarnecki et al.~\cite{czarnecki2008}, as part of their work in probabilistic FMs, proposed a visualization for recommending features during the interactive configuration of a product based on existing configurations. The visualization represents a score associated with each feature.

**Feature-based recommender system**: Regarding scores, Pereira et al. [2016b] proposed a *feature-based recommender system* to support the product line configuration process. Their main objective is to introduce an approach comprising an advanced feature-based personalized recommender system together with visualization and selection mechanisms[Pereira et al. 2016a ] to assist a user in the configuration process. They studied different algorithms to be used as scores for recommender systems in SPL configuration. They tuned the parameters of these algorithms automatically using empirical data and showed that the collaborative filtering algorithm, BRISMF (\textit{Biased Regularized Incremental Simultaneous Matrix Factorization}) outperformed the others.

We took this algorithm and we extended the visualization aspects which was very limited in both Czarnecki et al. and Pereira et al. works. In addition, both works did not consider NFPs as part of the configuration process.