Probabilistic Graphical Modeling for Software Product Lines

A Framework for Modeling and Reasoning under Uncertainty

by

Anas Almharat

A thesis submitted in partial satisfaction of the
requirements for the degree of

Doctor of Philosophy

Department of Computer Science and Informatics

In

School of Architecture, Computing and Engineering

of the

University of East London

Supervisory team

Dr. Rabih Bashroush

Dr. Usman Naeem

London 2016

Probabilistic Graphical Modeling in Software Product Lines

A Framework for Modelling and Reasoning under uncertainty

Abstract

This work provides a holistic investigation into the realm of feature modeling within software product lines. The work presented identifies limitations and challenges within the current feature modeling approaches. Those limitations include, but not limited to, the dearth of satisfactory cognitive presentation, inconveniency in scalable systems, inflexibility in adapting changes, nonexistence of predictability of models behavior, as well as the lack of probabilistic quantification of model's implications and decision support for reasoning under uncertainty. The work in this thesis addresses these challenges by proposing a series of solutions. The first solution is the construction of a Bayesian Belief Feature Model, which is a novel modeling approach capable of quantifying the uncertainty measures in model parameters by a means of incorporating probabilistic modeling with a conventional modeling approach. The Bayesian Belief feature model presents a new enhanced feature modeling approach in terms of truth quantification and visual expressiveness. The second solution takes into consideration the unclear support for the reasoning under the uncertainty process, and the challenging constraint satisfaction problem in software product lines. This has been done through the development of a mathematical reasoner, which was designed to satisfy the model constraints by considering probability weight for all involved parameters and quantify the actual implications of the problem constraints. The developed Uncertain Constraint Satisfaction Problem approach has been tested and validated through a set of designated experiments.

Profoundly stating, the main contributions of this thesis include the following:

- Develop a framework for probabilistic graphical modeling to build the purported Bayesian belief feature model.
- Extend the model to enhance visual expressiveness throughout the integration of colour degree variation; in which the colour varies with respect to the predefined probabilistic weights.
- Enhance the constraints satisfaction problem by the uncertainty measuring of the parameters truth assumption.
- Validate the developed approach against different experimental settings to determine its functionality and performance.

TO MY PARENTS

Contents

List of Figures

List of Tables

List of Abbreviations

ADL	Architecture Description Language
BBFM	Bayesian Belief Feature Model
BDD	Binary Decision Diagram
CVL	Constraint Variability Language
CSP	Constraint Satisfaction Problem
DAG	Directed Acyclic Graph
DM	Domain Model
FD	Feature Diagram
FM	Feature Model
FODA	Feature Oriented Domain Analysis
FORM	Feature Oriented Reuse Method
GP	Generative Programming
OVM	Orthogonal Variability Modeling
PLP	Product Line Practice
SEI	Software Engineering Institute
SPL	Software Product Lines
SPLE	Software Product Lines Engineering
UML	Unified Modeling Language

Acknowledgments

There are so many people to whom I owe credit for this thesis. This thesis, and the work it documents, would not have seen the light without the support and the encouragement from those who always had a faith in me.

During my PhD journey, I was blessed to meet new friends from all over the world. Those, who always were there for me and became my family; family to have, and friends to keep for lifetime.

If I may begin by acknowledging my director of study, Dr. Rabih Bashroush, who was always a great inspiration, and would always receive admiration for his hard work, leadership, bright ideas, dedication and encouragement, which he had been continuously providing during my studies. It has become clear to me that having him, as my first supervisor was invaluable bless.

I also would like to thank Dr. Usman Naeem for his incisive feedback during my research.

I would also extend my acknowledgements to Dr. Ameer Al-Nemrat, who was always a great support and pushed me to run the extra miles.

Finally, to my family back in Jordan, your unconditional love and support, had always motivated me to wake up early in the morning and do what I had to do, thank you!

"We can't solve problems by using the same thinking we used to create them"

- Albert Einstein

Part I

Motivation

Chapter 1

Introduction

1.1. “Discontent is the first necessity of progress”

Thomas Alva Edison (1913-1947), a great influencer and inventor once said: “Restlessness is discontent and discontent is the first necessity of progress. Show me a thoroughly satisfied man and I'll show you a failure”.

This lone quotation ignited the vital spark of my thesis, the same way discontent ignited revolutions, forward thinking and innovation as testified throughout history. Those words of sage ought to be consciously noted, for in them I find the spur of what is starting to take place in the realm of Software Product Line Engineering.

Linking this line of wisdom with Software Product Lines, the discontent with current feature modeling approaches and satisfiability techniques in software product lines, which are coupled with numerous problems, shaped the backbone in writing this thesis. The necessity to have more optimized feature models with better satisfiability techniques provided eagerness to writing this thesis.

1.2. Chaos isn’t just a theory; it is likewise a ladder

Chaos theory is a scientific theory best describing the unpredictability and randomness of systems as mentioned by the French mathematician Henri Poincare (1854-1912). It conveys that even negligible and unnoticeable variances in the beginning of any process might produce significant and weighty differences at the end. In terms of software product lines 'SPL' and Feature Modeling 'FM', any trivial error in the process can lead to vast complications and untraceable failures. Then again, every system is subjected to chaos. There's a reasonable probability and potential for any system, no matter how well designed, to fall into chaos.

Chaos is largely thought to be a drawback, obstacle and negativity. It is often related to mislead, vagueness and disorder. In this dissertation, the term 'Chaos' doesn't merely relate to the chaos theory, instead chaos takes after the lack of precision and the high level of complexity in terms of Data Modeling and automated analysis in software product lines. This Chaos ought to be terminated and advancements must be provided in view of software product lines. In terms of FM techniques, it implies that complexity must cease to exist; the upheavals present in feature modelling discussed in this dissertation are put under the lenses of study aiming to answer some questions and to find more suitable solutions.

1.3. Problem Statement

As stressed out beforehand, the province of feature modelling, and its automated analysis traverses great challenges and has considerable shortcomings.

In favor of tackling those shortcomings, this dissertation probes four questions scarcely taken into consideration by related literature.

Those questions are researched in depth in the following chapters:

- ***Probe question 1:*** What are the modeling techniques used to capture variability in Software product lines engineering?
- ***Probe question 2:*** How to quantify the occurrence uncertainty in model parameters, while maintaining the existing semantics?
- ***Probe question 3:*** To what extent can we enhance the visual expressiveness of a feature model?

- ***Probe question 4:*** How to improve the reasoning efficiency of constraints satisfaction problem, by taking into consideration the degree of uncertainty of model parameters?

1.4. Designated Approach

A concise view of the methods delivered in our dissertation is presented upon considering the following steps:

Step one: Exploration

The first step lays in launching and establishing a meticulous groundwork to the dissertation. This is pertained by means of extensive literature review, in which a state of art of software product lines is presented, with an emphasize on feature modeling to create a roadmap for the succeeding steps and derive queries to be systematically dealt with accordingly:

- An overview to understand software product lines and feature modeling techniques is targeted by investigating current approaches of feature modeling for Software product Lines 'SPL' from previous literature work.
- After providing a clear background, setting out definitions, and identifying the significance and substantiality of our addressed subject; an evaluation of feature models in software product lines is undertaken.
- This evaluation emerges into an in-depth study of feature modeling approaches by dismantling and analyzing them in practice, presentation and notations. Throughout this study, a real life exemplar developed by the author as an illustration and case to work on, is employed; which is the example of 3D Printer.

  The evaluation is undertaken according to designated criteria taking into consideration the existent glitches in selected feature modeling notations. The evaluation covers a selection of criterions: scalability, traceability, articulacy, comprehensiveness and visual presentation suitability.
- A systematic literature review is undertaken to provide a better understanding, identify knowledge gap, pinpoint limitations in current feature models, and detect the glitches and shortcomings associated. The knowledge gap lays in the truancy

of probabilistic feature models, absence of frameworks to capture uncertainties in feature models of software product lines, and the lack of reasoning techniques.
- The overall review derives and leads to dealing with the research questions answered using methods elaborated in the following steps.

Step two: Construction

The second step takes place as an attempt to find solutions to the glitches present in current feature models, address the knowledge gap, and answer the research questions resourcefully. This step show casts the opportunity of probabilistic modeling and the potentials for evolving the current feature models by addition of probabilistic weighting to quantify the existing uncertainty.

- An exploration of other relevant domains (machine learning, Data modeling, and Bayesian belief networks) to investigate potential techniques that can fill up the knowledge gap is undertaken.
- This leads to find inspiration and identify adequate techniques to employ in solving problems and build up the evolved feature model.
- Bayesian belief network tends to be compatible; consequently it is set as the approach to probabilistic modeling and the creation of Bayesian Belief Feature Model (BBFM).
- Mathematical studies are commenced to capture semantics of the feature model and reason about it.
- The proposed solution manages to exploit the problem, capture uncertainties and quantify them. Consequently, semantic mapping is observed to capture the notion of the feature model which facilitates the development of the BBFM.
- Subsequently, we form an approach for weighting parameters, features and dependency flows among features. This approach studies various scenarios in which mathematical analysis is applied to prove each theorem and validate it.

Step three: Substantiation

This step wraps up our approach to provide authentication and validation of the work submitted in the first and second step.

- This step addresses and show casts the findings, in terms of solving the bridges and providing an enhanced evolved feature model, in which the feasibility and practicality of the solutions are pointed out.
- In order to demonstrate the outcomes, we provide a graphical presentation of our feature model, which augments present feature models graphical presentation in terms of visualization.
- We use gray scale mapping to better visualize the probabilistic value assigned to each feature. Accordingly, shades of gray based on different weighting schemes are assigned, such that the distinct weighting value is translated by the intensity of the shade of gray.
- According to the assigned values in the weighting theorem, our work is extended and developed to reason about uncertainties. Besides, the efficiency; our proposed technique is scrutinized in terms of reducing problem size and reasoning time.
- This step finds closure by discussing the obtained results and forecasting any future work or improvements.

1.5. Goals and Contributions

The dissertation intends to provide insightful exploration of the existing feature modeling techniques in software product line engineering 'SPLE'. Throughout a systematic review, we were able to define, identify and analyze the current approaches used to manage the variability in SPLE. After examining the current practices of SPLE data modeling and reasoning, we were able to identify the knowledge gap as follow;

"To the best of our research we argue that; SPLE lacks to:

1. Probabilistic quantification of the data model parameters.
2. Actual quantification of features implications.
3. Framework to anticipate the degree of uncertainty of the satisfaction problem.
4. Mathematical approach to tackle the uncertainty problem and reason about it."

To overcome the aforementioned gap of knowledge, this thesis was designed to:

1. Provide a comprehensive mathematical framework, to quantify the uncertainty measure of model parameters.

A probabilistic modeling approach was developed to capture the dependency semantic among model features, while assigning probability weight for all features and dependency functions.

2. Employ the predefined probabilistic weights to enhance the visual expressiveness the developed notation, by a sensible integration of the colour use.
3. Utilize the predefined probabilistic weights to improve the constraint satisfaction process, by emphasizing the uncertainty measure of the problem space. Subsequently, develop an algorithm to satisfy the model constraints under uncertainty, and to anticipate the probability of obtaining a satisfiable configuration amongst the problem space.
4. Experimentally validate the aforementioned findings. Moreover, we were able to exploit the satisfaction problem behavior, allowing better understanding of the constraints actual implications, and enlighten the possible techniques to improve the reasoning process during the early stages of the model design.

1.6. Reader's Guide Map

In this section, we establish and provide a guide map to the thesis for the ease of readers. We start our dissertation by presenting a state of art for the variability modeling in software product lines, the advances, knowledge gaps, and challenges discussed in previous work. The Literature review will put an emphasize on feature modeling notations of SPLE and its glitches to give a roadmap for the contribution. The research questions proposed above are to be systematically dealt with throughout the review.

The subjects covered are systematized as follows:

- Chapter 2 "*Background*" gives a preface to the notion of Product Line Engineering 'PLE'. It provides a background on the emergence of SPL, and the motives of its emergence. It discusses SPLE in depth, provides a concise chronicle on the attempts to advance in SPL development through software modeling, and offers an insight to its benefits and limitations.
- Chapter 3 "*Managing Variabilities*" handles thoroughly the variability modeling terminologies, principles and applications. It presents the types of variability modeling in SPL.

- Chapter 4 “*Feature Modeling in Depth*” includes a critical review on features and feature modeling notation and investigates the previous approaches to FM using a scenario of 3D printer. This followed by a summary of the limitations and challenges within feature modelling.
- Chapter 5 “*Modeling under Uncertaint*” will provide a comprehensive framework for quantifying the uncertainty measure in SPL. The obtained measurements are later used to structure Bayesian Belief Feature Model, with emphasis on the visual expressiveness of the developed model, throughout “use of colour”.
- Chapter 6 “*Reasoning under Uncertaint*” will present the designed method to tackle the *Constraint Satisfaction Problem* while taking into account the recomputed uncertainty measures. To validate the developed algorithm, experiments were conducted in order to validate the functionality of the proposed approach. This is then followed by an extensive discussion of the results.
- Chapter 7 “*Conclusion*” section concludes the findings and identifies improvements and future work

"Learning is more than absorbing facts, it is acquiring understanding"

-William Arthur Ward

Part II

Introduction

Chapter 2

Background

Software Product Lines 'SPL' is a favorable and providential paradigm for the progression and prosperity of creating methodical and intensive software systems (Clements, 2002). It aims to provide an organized SPL, in which it furnishes a collection of products that shares lots of commonality rather than variability (Benavides, Segura, & Ruiz-Cortés, 2009). In other words, SPLE brings about the commonalities and variabilities in a set of reusable assets to provide a flexible efficient system with an ease of use, management, configuration, and customization (Bachmann & Clements, 2005; Van Gurp, Bosch, & Svahnberg, 2001). FM is a notation that illustrates the SPL; it represents the Software products as set of features. It is an information model displaying all core assets and variables that a customized software product can require (Rincón, Giraldo, Mazo, Salinesi, & Diaz, 2015).

Because of the high complexity of feature models, it was proven that those models are vulnerable and open to errors, complications and faults (Czarnecki & Wasowski, 2007; Lee, Kang, & Lee, 2002; Thüm et al., 2012; Thüm, Batory, & Kästner, 2009; White et al., 2010; White, Dougherty, Schmidt, & Benavides, 2009). Consequently, the automated analysis of feature models derived as a support to cope with the challenges of FM.

Numerous works conveyed in the literature took in hand the identification of shortcomings and faults in FMs and their automated support (Arcaini, Gargantini, & Vavassori, 2015; Benavides, Segura, & Ruiz-Cortés, 2010; Gargantini & Fraser, 2011).

However, only few of these tenders are able to support a rigid account on dealing with defects and complexities.

This part of the dissertation provides an in-depth literature review on the software product lines followed by an insight on feature modeling. It scans through the work of previous pioneers in the province of feature modeling to assemble all the approaches mentioned in order of bestowing a solid starting point for this dissertation and ridding bias and misperception.

2.1. Software product lines: pitfall of economy and the rise of PL

2.1.1. The Pitfall of Economics

According to Oxford dictionary, Economics is defined as the area of knowledge concerned about the production, distribution and consumption of products and services and the supply of money. It is the deliberation of the way choices are made under circumstances of scarceness and resource limitations, and the results of those choices on society. These definitions are to be very broad and generic.

When talking about the scale of economics, economics can be discussed on the basis of two ranging: Wide and narrow ranging (Boehm, 1981).

Macroeconomics resembles the wide range of the economics study; it studies the economics and decision made in scarcity of resources on a global scale. It takes into consideration the effects and subsequence of those decisions on matters such as trade policy, interest and tax rates. However, Microeconomics takes after the narrow range of the economics study; it studies the decision made in scarcity of resources on a personal and subjective scale. It takes into concern the effects of those decisions taken, either as a person or as an organization, on matters specific attributes such as cost and insurance.

When talking about Software Products economics, it is deduced that it falls in the micro economical scale. It is directly related to decisions taken from SPLE (Da Silveira, Borenstein, & Fogliatto, 2001). These decisions are taken in scarcity of resources since there’s always deficiency in time, money and feature selection. Moreover, critical limitation in resources can be present in terms of computing capabilities such that the software product features gets affected.

2.1.2. Mass Customization Notion

Mass customization in a broad-spectrum, is related to the capability to deliver customized products or services through flexible and dynamic procedures in high capacities coupled with cost efficiency (Da Silveira et al., 2001). The Notion arose in the late 80's and was considered as a natural and expected trail to the procedures that became high in flexibility and enhancements as regards the quality, quantity and costs (Lau, 1995). Besides, mass customization appears as a mean for companies to step out in a time of challenges and competitiveness (Kotler, 1989). Furthermore, Mass Customization was termed as the capability to supply exclusive and individually-tailored products or services through optimal time, integration and manageability (Davis, 1989; Eastwood, 1996). Figure 2. 1 presents an example of mass customized vases that varies from one another, and is tailored each exclusively.

Mass customization in a narrower spectrum, is a flexible process that organizes and manages information technology and assets to bestow an extensive choice of products or services in accordance to specific and precise requests of individual customers, at a cost proximate to the one of items that aren’t specified according to personal needs (in other words, mass produced products without individualization) (Hart, 1995; Joneja & Lee, 1998; Kay, 1993; Kotha, 1995; Ross, 1996).

Figure 2. 1 Customized 3D printed vases

The intent of mass customization is based on numerous ideas some of them are listed as follows:

- Deliver high diversity of products at the minimal cost by means of flexibility in production and information technologies (Åhlström & Westbrook, 1999).
- Response to the Growth in demand for product diversity and individualization by customers (Kotler, 1989).
- Provide products which are more personal and meaningful, more appreciated by customers.
- Present products that are feasible and practical in terms of planning, design, production, distribution, and facilitating (Hart, 1995).

2.1.3. Mass Customization and Mass Production

Going through the literature, the topic of the correlation between mass production and mass customization was always stressed out. The debate laid emphasis on whether mass production and mass customization belonged to the same continuum or they were independent of each other, taking into account, their functional and conceptual differences and the likelihood of using both mass production and mass customization at the same time.

On one hand, Authors including Lau (1995) sees that mass customization and mass production belongs to the same continuum. They suggest that in order the mass customization approaches to be effective, it must have a background on mass production systems.
On the other hand, Pine, and Boynton (1993) see that mass customization and mass production are two different entities and they can't be at the same continuum. They have different goals, approaches, and concerns. When talking about mass production, it is noticed that the first concern is the product itself, preceded by the process of production. However, Mass customization is concerned mainly about the process that allows the diversity of products. Figure 2. 2 exemplifies the different dialogues of mass production and mass customization. It shows that mass production is a one way dialogue, whereas mass customization adapts an interactive progression dialogue.

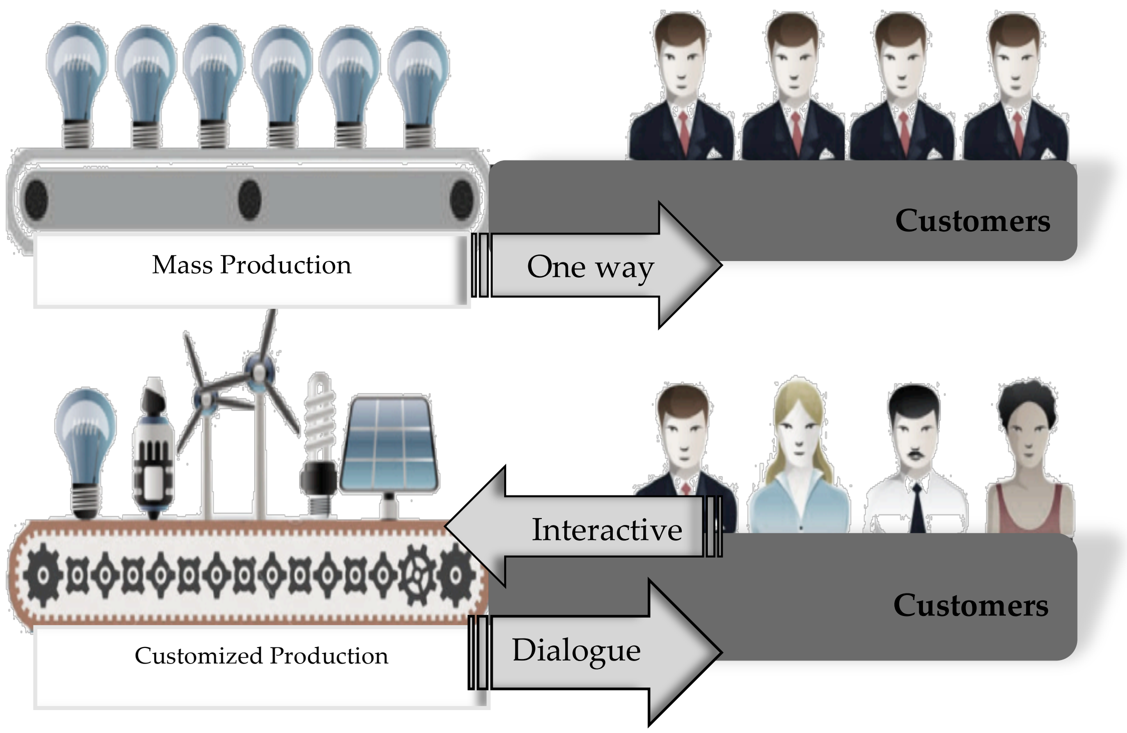


Figure 2. 2 Mass production versus Mass customization dialogues

2.1.4. Mass customization Essentials Kit

- Technology must be accessible and manageable; the employment of advanced production machineries, technologies and tools is essential to enable to hand out mass customized systems (Adamides, 1996; Hirsch, Thoben, & Hoheisel, 1998; Kotha, 1996; Lau, 1995). Flexible and well managed Information technologies process is a vital part for the customization of products.
- Products should be adaptable to alteration. Effective mass customization products must be modularized, flexible, and constantly improved. Moreover, mass customization processes need prompt product development and innovation competences (Pine, Victor, & Boyton, 1993).
- Mass Customization system must have the ability to interpret new demands into innovative products. This necessitates the development of flexible networks (Pine et al., 1993), besides the production engineering knowledge and expertise, and process technologies (Kotha, 1995).

2.2. The Prominence of Software Product Lines

Software Product Lines gained attention after the emergence of software reuse succeeding the blooming of mass customization. The idea of SPL, in other words, reusing software for creating bespoke products (Kang, Cohen, Hess, Novak, & Peterson, 1990; K. Pohl, Böckle, & van Der Linden, 2005), was an innovative and pioneering idea which arose as an alternative to the conventional method of production (Eriksson & Hagglunds, 2003; Kotler, 1989). The conventional method of software production was based on individual system development (Åhlström & Westbrook, 1999). Contrariwise, SPL transit production to another broader level; it is based on families sharing common functionalities (Bosch & Bosch-Sijtsema, 2010; L. M. Northrop, et al.). As a matter of fact, Grouping those systems with common functionalities and producing SPL instead of developing each system from beginning; prove to be more appealing and desired for the industry (Clements, 2002; Gomaa, 2005; Van der Linden, Schmid, & Rommes, 2007). Figure 2. 3 demonstrates the transition from conventional production to customized product lines which provided a broader range of alternatives.

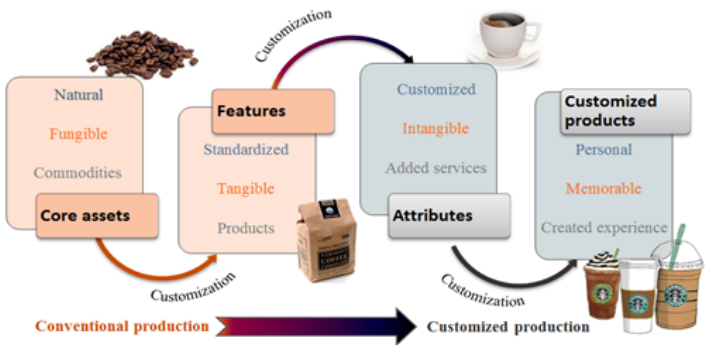


Figure 2. 3 Transition from conventional production to product lines

2.2.1. SPL: Why and Wherefores

The reasons of the prominence of product lines cannot be quantified given that it is a vast discussion. However, in accord with previous literature (Clements, 2002; Cohen, 2003 2005, Mazo et al., 2008), statistics, and case studies (Clements, Cohen, Donohoe, & Northrop, 2001), the main incentives and elements are synopsized in Table 2. 1 Organizational and Business Advantages of PLs.

2.2.1.1. *Advantages of software product lines: Epigrammatic list*

Organizational advantages of Product Lines

	Tangible Advantages	
⇗	Profitability	Further market agility and market shares are reached due to product line approach. Moreover, Market presence is maintained and competent growth is sustained (L. M. Northrop, et al.).
⇗	Quality	Product defect density is highly reduced in Product lines. Also resolving those defects and redundancies is mostly unproblematic (Van der Linden et al., 2007)
⇗	Performance	Performance is significantly improved through prompter dealing with algorithms and variation and circumventing timing problems (Clements et al., 2001).
⇘	Time to Market	Time to field and to launch is reduced, because of the reuse of assets and diminution of replicated mistakes and deficiencies in the system (Heymans et al., 2008).
⇗	Productivity	More flexibility in meeting customer's demands and more ease in amendments and modifications. Product line Assets are meant to be easily implemented, and thus regarded as commercial off-the-shelf products (Thao, 2012).
⇘	Code Volume	Source code size and the quantity of design objects for subsystems in product line systems are lessened in comparison to that of the traditional single systems (Bosch et al., 2001).
	Intangible Advantages	
⇗	Software Developer acceptance	The product lines tends to expect satisfaction and certitude from the developer for the system as groundwork as well as the approach itself (Clements, 2002; L. M. Northrop, et al.).
⇗	Professional satisfaction	As the monotonous repetitive tasks that presented in conventional systems are sidestepped by the reuse of assets in Product lines System, The main focus is redirected to more

		challenging, mission-specific requirements or on performance adjusting and perfecting. Henceforth, professionals are more contented (L. M. Northrop, et al.) .
↘	Attrition rate	Staff resignation, turnover and renewal rate of staff is assumed to be lower whilst adapting product lines systems, in comparison to the conventional systems (Clements et al., 2001).
↗	Customer satisfaction	Customers develop more sense of attachment; content and comfort since product lines offers a more predictable approach with less redundancies rates and better quality products (Heymans et al., 2008).
Businesswise Advantages of software product lines		
↘	Production and maintenance costs	Using same approach, process, tools and techniques and with less redundancies and complications production and maintenance costs are relatively reduced. Compatibility of objective system and products with evolving capabilities is ensured. Wider interoperability and flexibility, before executing subsystem and device production, are guaranteed (Ardito et al., 2011).
↗	efficiency in the processes	Product lines promote improving the consistency and reliability of the user interface. In addition, it offers more efficient integration of the products by the use of common standards and products to meet training, and test requirements (Clements et al., 2001).
↗	budget and time planning	Product lines implementation provides a permanent integrated and interoperable infrastructure, and thus it causes decrease in risks, costs and schedule time for planning and preparations (Heymans et al., 2008).

Table 2. 1 Organizational and Business Advantages of PLs

2.2.1.2. *Advantages of software product lines: profound list*

- **Upturn of Quality**

The common assets of the production line craft the core of the entire software family. Those common assets are subjected to review, checking and testing in various products to prove their optimal performance (Ebert & Smouts, 2003). The manifold and extensive testing of common assets in the SPL leads to detecting faults, oversights and defects (Figure 2. 4).
Consequently, all products of the software line are of a better quality (Lang, 2015; K. Pohl et al., 2005). Figure 2. 4 evidences the upturn of quality in SPL, as a result of the decrease in defects.

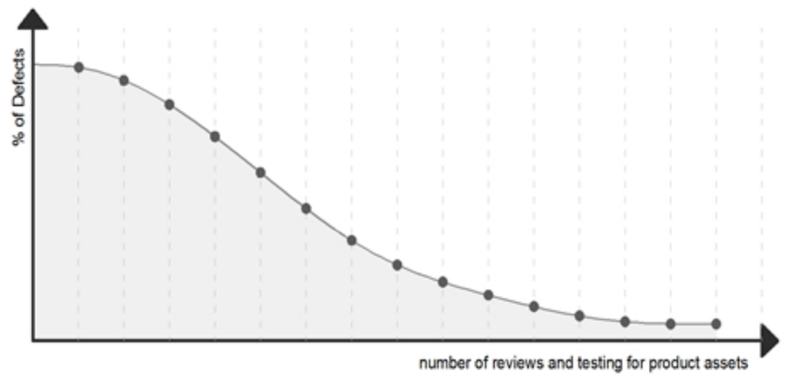


Figure 2. 4 Percentage of defects per number of reviews and testing (Ebert & Smouts, 2003)

- **Ease of upgrades**

The addition of new features, Alteration of existing features, amendment and upgrading of assets in the product line provides an undemanding opportunity to upgrade the software product line and its whole derived products.
In comparison to conventional single system productions, SPL has definitely more ease of upgrades and less effort in making alterations (Ebert & Smouts, 2003; K. Pohl et al., 2005).

- **Cost Efficiency**

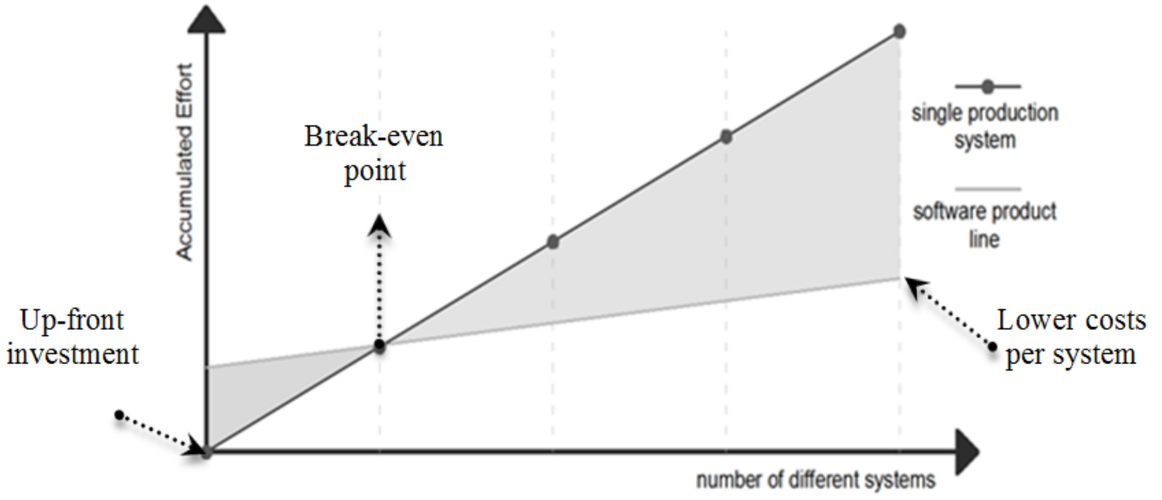


Figure 2. 5 Costs of SPL vs cost of single production system (Ebert and Smout, 2003)

To a large extent, cost efficiency is one of the foremost motives for commencing software product line engineering, as it allows creating solutions with fewer expenses and more profits. The reuse of assets for multi production through a single software product line system implies a significant cost reduction (Mazo & Salinesi, 2008).
Although SPL requires an upfront investment for planning and setting strategies, in the long run the reuse of assets gives Software product line advantage over single production systems in such a way that it has lower cost as the number of products increase (Figure 2. 5).

- **Time Saving**

SPL lessens the time needed to launch the product in the market considerably. In comparison with the conventional production systems, the latter system time to market was approximately constant. Contrary, albeit at the outset product line systems takes longer time to arrange and plan common features, the launching time in market is cut down gradually as the number of systems and developments increase due to reuse (Mazo & Salinesi, 2008) (Figure 2. 6).

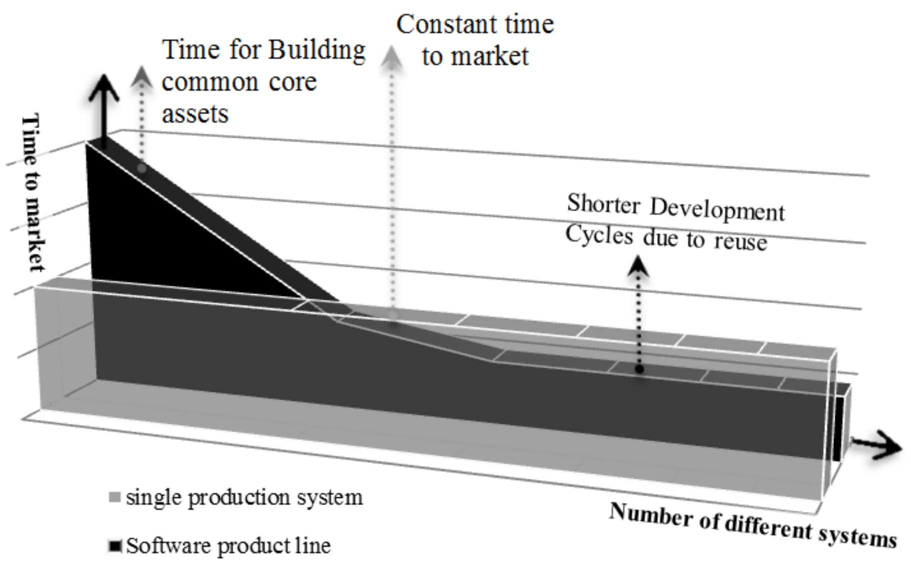


Figure 2. 6 Time to market for SPL vs single production system (Ebert & Smouts, 2003)

- **Leadership Supremacy**

SPL unlock cutting-edge opportunity gateway in competitiveness, innovation, productivity and profitability (Ardito et al., 2011).
It opens up new possibilities that don't exist in conventional single production systems. Accordingly, Leadership supremacy is achieved by the mean of SPL that promotes delivering large number of customizable products in less time and effort, and higher qualities (Figure 2. 7).

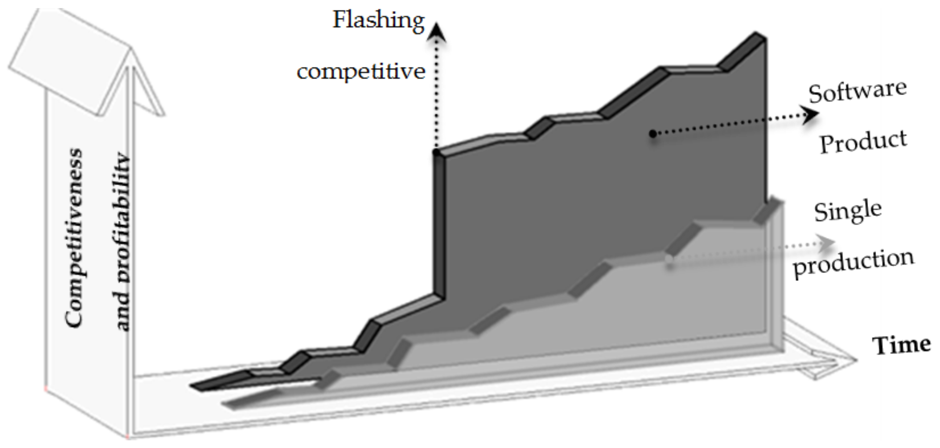


Figure 2. 7 Productivity of SPL vs productivity of single system (Biglever software inc)

2.2.2. Myth versus Reality: SPLE as Lego construction

2.2.2.1. *The myth*

"And then we'll be able to construct software systems by picking out parts and plugging them together, just like Legos…" (Shaw, 1998). The myth states that the archetype model of reusability and the best resemblance to software product lines is as simple as "Lego" (Crnkovic & Larsson, 2002).

2.2.2.2. *The Fact*

It's much more complicated and contains lots of incompatibility and bugs. It is consisted of a complex system with specific function and characteristic specific compatibility (L. M. Northrop, 2006). Figure 2. 8 iluustrate the difference between the myth and the reality of SPL modeling, in terms of complexity.

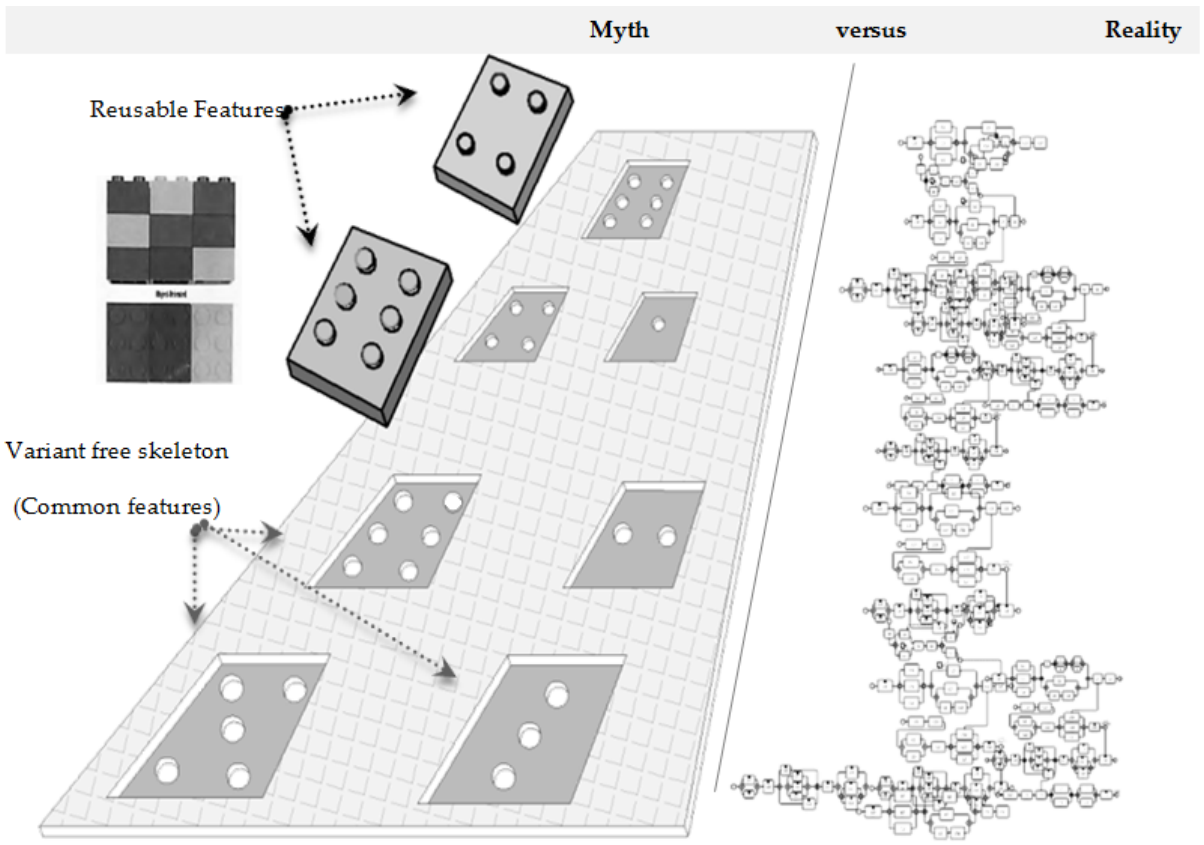


Figure 2. 8 SPL Lego myth (Inspired from Crnkovic et al., 2002 ; Mohabbati, 2013)

2.2.3. Drawbacks of SPLE

After noting that software product lines isn't an easy fix-all solution or as easy as a Lego plugging game, it's important to mention the costs and draw backs of SPLE. The table below (Table 2. 2) points out some of the common disadvantages of SPLE:

Disadvantages of software Product Lines		
⇗	Cost of developing core assets	The development of the common core features in the product lines is costly and requires upfront investment (Bosch et al., 2001). For the upfront investment money to pay off It usually needs at least two to three products to be built as a family (Mazo & Salinesi, 2008; L. M. Northrop, et al.) (Figure 2. 5).
⇗	Training staff	The training of staff in the new way of doing business is considered to be an expensive mission (L. M. Northrop, et al.). Those staff must not only be trained in software engineering but also in corporate procedures to ensure that the product line practice can and will be used in accordance with the current process. Staff must be specifically trained for the product line and new training materials must be created to address the product line which all requires more costs (Eriksson & Hagglunds, 2003).
⇗	institutionalizing	A risk associated with the institutionalizing of a product line approach is resistance from personnel to the new way of doing business. This type of resistance is often found in the middle level management (L. M. Northrop, et al.) and might require those persons to be reassigned to other tasks.
⇗	Marketing and sales support	The success of SPL relatively depends on marketing and sales support. The company must invest in long-term sales plan in order to succeed (Ebert & Smouts, 2003).

Table 2. 2 Drawbacks of SPLE

"Employ your time in improving yourself by other men's writings, so that you shall gain easily what others have labored hard for."

-Socrates

Part III

Literature Review

Chapter 3

Managing Variabilities

It is essential to unseal the paradigm of software product lines in sequence of breaking it down into segments, to promote better comprehension of all its facets. This chapter will approach variability modeling, the focal facet of software product lines, in all its associations, foundations and applications.

3.1. SPLE: a tale of Variabilities

Software requires to be tailored to changeable requirements, since the customer demands, markets, and hardware are changing and expanding (Muschevici, Proença, & Clarke, 2015). If each and every altered product is handled individually, the overhead of dealing with all the alternatives drastically converts to an infeasible option (K. Pohl et al., 2005). As a solution for this dilemma, Software Produce Line Engineering is present (Apel, Batory, Kästner, & Saake; Czarnecki & Eisenecker; K. Pohl et al.).

Expounded by Bosch (2001.), Clements and Northrop (2001), and Pohl, Bockle and Van Der Linden (2005), software product lines are basically an assembly of software products holding mutual set of features that comply with the requirements of a specific domain. SPL's outcomes are delimited and implemented as a combination of common and variable features subsequently bringing out the final software products (Muschevici et al., 2015). A feature consequently is a property or functionality used to capture commonalities or discriminate among systems in SPL (Czarnecki & Eisenecker, 2000). Variability modeling is an important method to present and represent common and variable features, and to decide which features to be supported in a product line and which ones to be deserted (Glück & Lowry, 2005).

SPL are practically multifarious systems formulated from a treble process (Clements et al., 2001; K. Pohl et al., 2005), Listed as follows:

3.1.1. Domain engineering

Also known as family engineering or core asset development (Heuser & Pernul, 2009), the domain engineering phase is responsible for the production of software core assets to be used in different products of the SPL. As mentioned by Pohl et al. (2005), and Clements and Northrop (2001), the domain engineering goals lays as follows:

- Identifying the commonality and variability among the whole elements of the SPL.
- Defining flexible architecture that addresses the commonalities and variabilities.
- Establishing the set of application that the SPL is planned for.
- Modeling and defining the scope of the SPL.
- Constructing and implementing the reusable assets that leads to the desired variability in application engineering.

3.1.2. Application engineering

Also known as product development or product derivation, the application engineering phase is responsible of deriving products according to specific combination of features which is based on the foundation of commonalities in addition to variable asset selection (Clements et al., 2001). The goals of the application engineering are stated as follows according to Pohl et al. (2005):

- Attaining a high reuse of the domain assets.
- Utilizing the commonalities and variabilities.
- Adjusting variabilities in consistent with the needs of the application.
- Producing individual systems from core assets in accordance to individual needs.

3.1.3. Management

Imperative to both processes (domain and application engineering), is the management of variability across the product line (Halmans & Pohl, 2003), in which resources are given, coordination is assured and supervision on both domain, and application engineering process is ensured (Benavides, Segura, Trinidad, & Cortés, 2007).

Figure 3. 1 clarifies and sheds the light on the process of SPLE as demonstrated below:

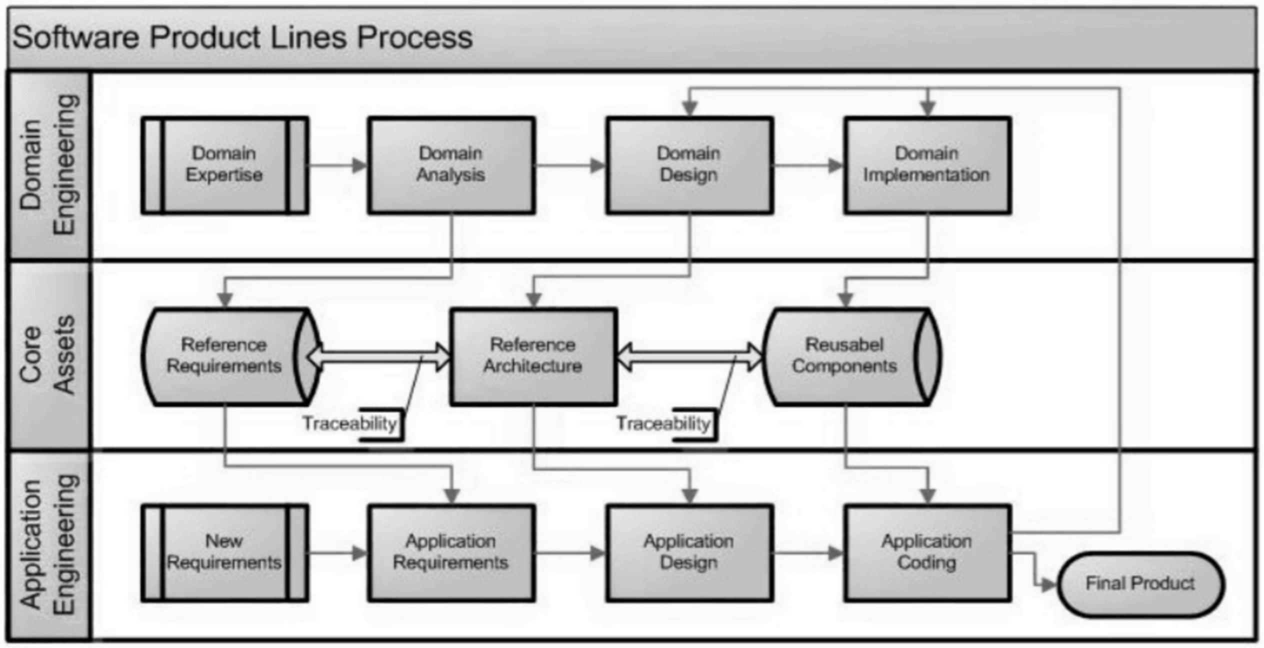


Figure 3. 1 Scheme of Software Product Line Engineering (Seke, 2013)

3.2. Commonalities and variabilities

As stressed out earlier, domain engineering and application engineering set the foundation and groundwork for the software product line engineering. SPLE is in charge of the variability management, in other words the process of identifying and sorting out commonalities and systematizing the variabilities of software artifacts and models (Berg, Bishop, & Muthig, 2005; L. Northrop & Clements, 2001).

Specialists in the province of SPLE, such as (Knauber, Muthig, Schmid, & Widen, 2000); (Macala, Stuckey Jr, & Gross, 1996); (Coplien, Hoffman, & Weiss, 1998); (Ahmed & Capretz, 2011) lay emphasis on the essential role for the SPL in terms of taking in hand the commonality and variability in the development of products. The scrutiny of commonalities and variabilities provides the software engineers with an organized methodical tactic of conceptualizing and pinpointing the product family they are generating (Coplien et al., 1998).

Albeit a unique perception, characterization or meaning of variability is not distinguishable, Literature points out several definitions of commonalities and variabilities. This section will highlight abridged notions of them:

For instance, Coplien et al. (1998) consider the variability in SPL is the supposition of how assets in product families can differ from one another. Henceforth, in their standpoint, variability postulates the distinctiveness of a product line system in accordance to certain expectancies of a customer. On the other hand, commonality deals with the suppositions that are constantly existent in each product of the software product line.

Svahnberg, Van Gurp, & Bosch (2005) outlines variability in terms of Software product lines perspective, adopting the perception of variability as the capacity of a system or product to be dynamic in terms of having the capability of proficiently changing, extending, and being adaptable to alteration, customization and configuration for specific uses in specific conditions and contexts.

According to Bosch et al. (2001), there are two different approaches to variability, first approach as a subject, and the second as an object.

Variability as a subject is an assorted entity or a diverse property of this entity. However, variability object is a precise illustration of a variability subject. The variability object is managed to identify the different approaches in which the variability subject can diverge.

For an instance, given the example in section 3.3, when considering 3D printers (see Figure 3. 4), the variability subject may well be the model making, and a variability object might be the diverse varieties of model making such as material, resolution and volume.

Not only does the literature point out different definitions of commonalities and variabilities, it also speaks about different categories of them;

Svahnberg et al. (2005) consider variabilities are made of five different stages, Bachmann et al. (2005) suggest another distinct categories of variabilities. in addition other researches categorizes variabilities by the notion of time and space (K. Pohl et al., 2005; Van der Linden et al., 2007), or according to essential and technical categories (Halmans & Pohl, 2003), externality and internality (K. Pohl et al., 2005), and finally according to Software product line engineering (Metzger, Pohl, Heymans, Schobbens, & Saval,2007).

Undoubtedly, research is challenged by the management of all the commonalities and variabilities of product lines; nevertheless, variability modeling tends to be the favorable option to facilitate their management (L. Chen, Ali Babar, & Ali, 2009).

Various approaches are suggested and put in use throughout the study of SPL to tackle the challenge of variability, which are to be explored in the following section 3.3.

3.3. Modeling Software Product Lines' Variables

In SPLE, it is evident that the domain engineering process is responsible of identifying the commonalities and variabilities of the product line, developing the core reusable assets, as well as modeling the variabilities of the SPL (K. Pohl et al., 2005), and thus requires more effort (see Figure 3. 2).

Variability modeling is a significant mean that allows interpretation and perceiving commonalities and variabilities in SPL, in addition to sustaining product customization and derivation. Amongst the numerous approaches of variability modeling identified in literature, Feature modeling 'FM' and decision modeling 'DM' are considered to be the most important approaches, in addition to Orthogonal variability modeling and others (see Figure 3. 3).

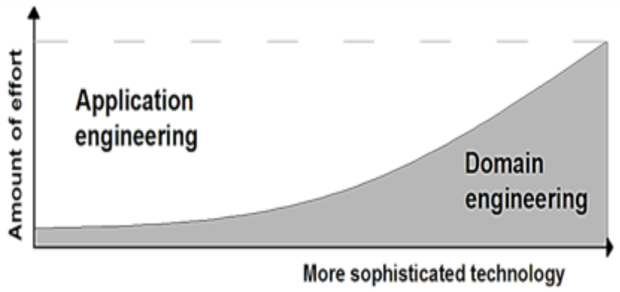


Figure 3. 2 Domain engineering versus Application engineering (Deelstra et al., 2004)

To outline our approach, in this section a suggested motivating exemplar of 3D printer product line is proposed and presented throughout the review. Based on this exemplar, different types of variability modeling are explained.

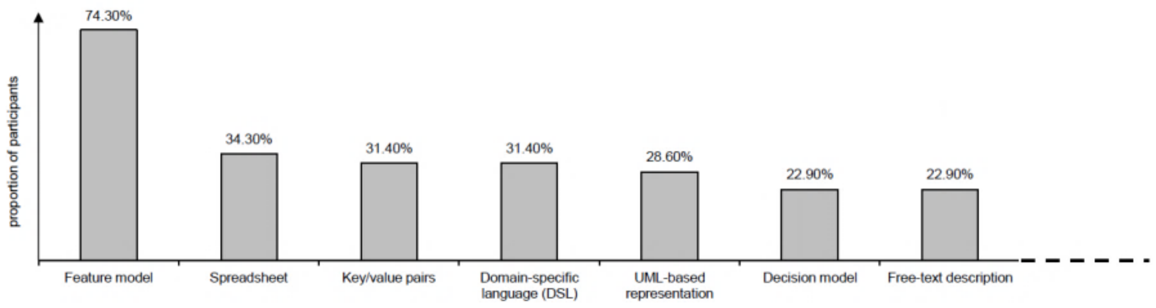


Figure 3. 3 High heterogeneity of variability modeling notations (Berger, 2013)

3.3.1. Feature Modeling

Feature modeling is the most famous approach to modeling variability present in literature (Czarnecki & Kim, 2005; Griss, Favaro, & Alessandro, 1998; Kang et al., 1998; Männistö & Bosch, 2004; Riebisch, 2003; Schobbens, Heymans, & Trigaux, 2006).

Chapter "Feature Modeling in Depth" in this dissertation is dedicated to the enlightenment of feature modeling in depth.
In brief, feature model presents the whole possible facets of a product, and thus represents a diverse product line system. In other words, feature model aims to identify all possible features and their possible relationships and constraints (Benavides et al., 2010).

Figure 3. 4 is a proposed exemplar that portrays a suggested simplified feature model driven from the inspiration of the 3D Printers. The model illustrates the approach of feature model for modeling the variabilities of 3D printers and listing the common features.
Based on the proposed model, all 3D printers include a software, body, model making abilities and connectivity tools. However, accessories are optional and their presence varies on demand. Moreover software in the printer can vary as there is a number of software to choose from, and include or exclude.

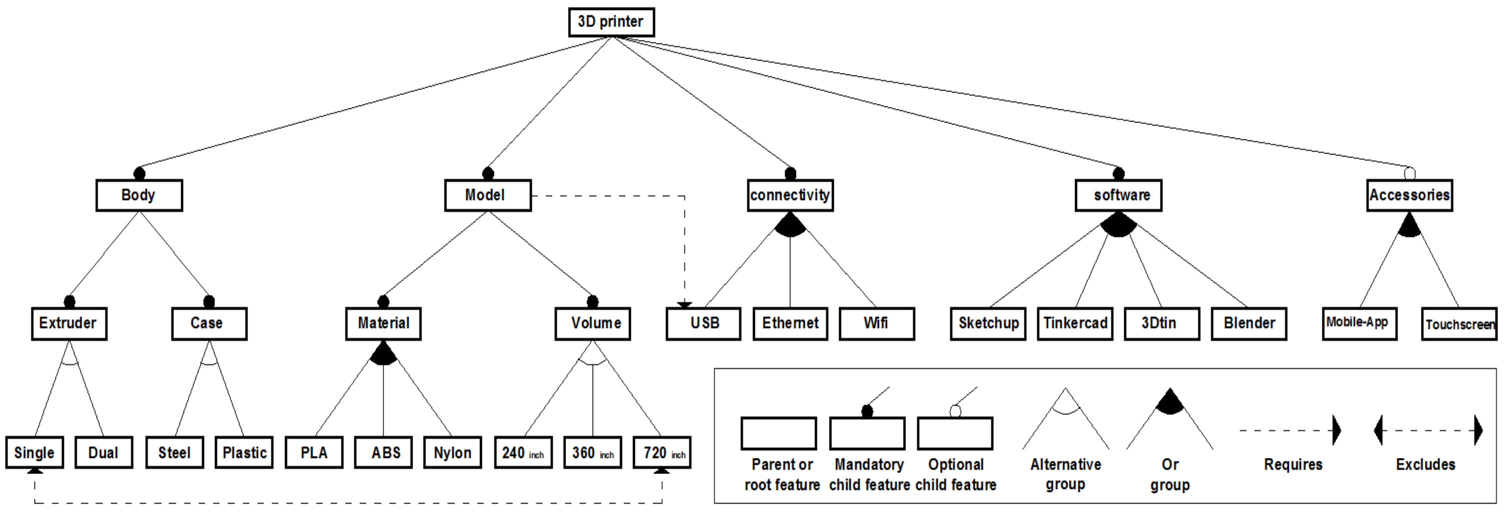
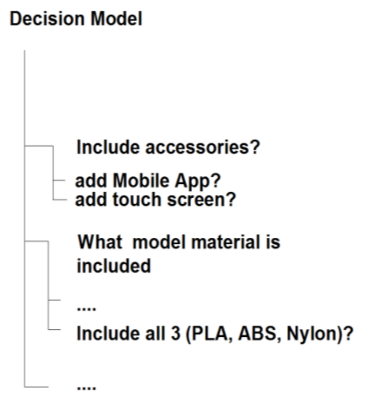


Figure 3. 4 Suggested Feature model for 3D Printer

3.3.2. Decision Modeling

Alongside the Feature modeling approach, exist the Decision modeling approach, which is also widely used to model variabilities. A decision model is a support outlining the decisions needed to be done in order to identify an element in a specific domain (Bézivin, 2001). Decision models are documents presenting decisions, their attributes, and dependencies (Atkinson, Bayer, & Muthig, 2000). Consequently, decision models brings about the diverse products by putting values to the decisions through setting out questions and relating them with possible answers; as a response of the arrangement of decisions dependencies.

Figure 3. 5 depicts a proposed decision model for the exemplar of the 3D printer, in a textual and a tabular representation.



Entity	Decision question	range	cardinality	constraints
accessories	do you want accessories	yes/no	1	N/A
Model	do you want accessories	yes	1	Model: all requires Usb connectivity. yes
extruder	what type of extruder do you want?	single/dual	1:1	Extruder: single excludes 720 inch volume. yes
case	what type of case do you want?	steel/plastic	1:1	N/A

Figure 3. 5 Examples of Decision modeling for 3D printers

3.3.3. Orthogonal Variability Modeling

In addition to the Feature modeling and decision modeling, there's also the orthogonal variability model that is used to model variability descriptions. The orthogonal variability model is presented by a graphical notation which describes the variability points, variants and their dependencies (K. Pohl et al., 2005).

Figure 3. 6 presents a simplified orthogonal variability model approach of the proposed exemplar of the 3D printer.

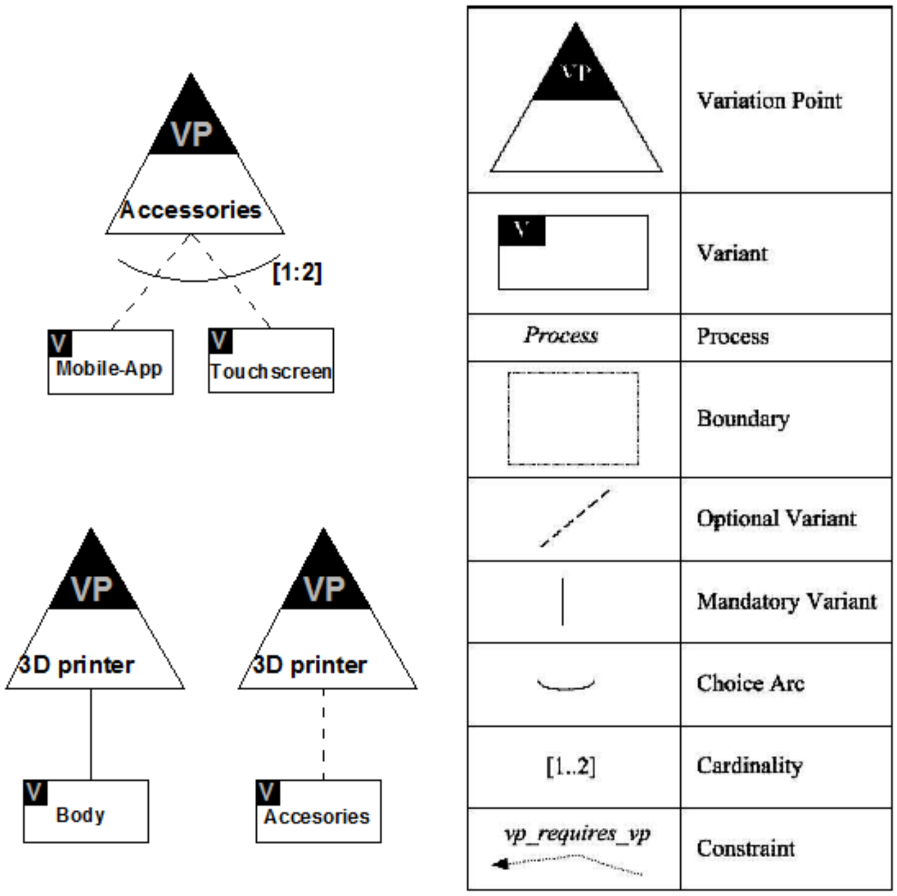


Figure 3. 6 Suggested Orthogonal variability model for 3D printer

3.3.4. Other Modeling Approaches

Variability model approaches are not limited to the feature, decision and orthogonal variability model, instead it extends to include approaches such as UML based variability model, and ADL based variability models, Constraints variability languages CVL, COVAMOF, and ConIPF (Haugen, Moller-Pedersen, Oldev, Olse, & Svendsen, 2008; Sinnema & Deelstra, 2008).

Figure 3. 7 presents a simplified UML Use Case diagram for the suggested 3D printer variabilities.

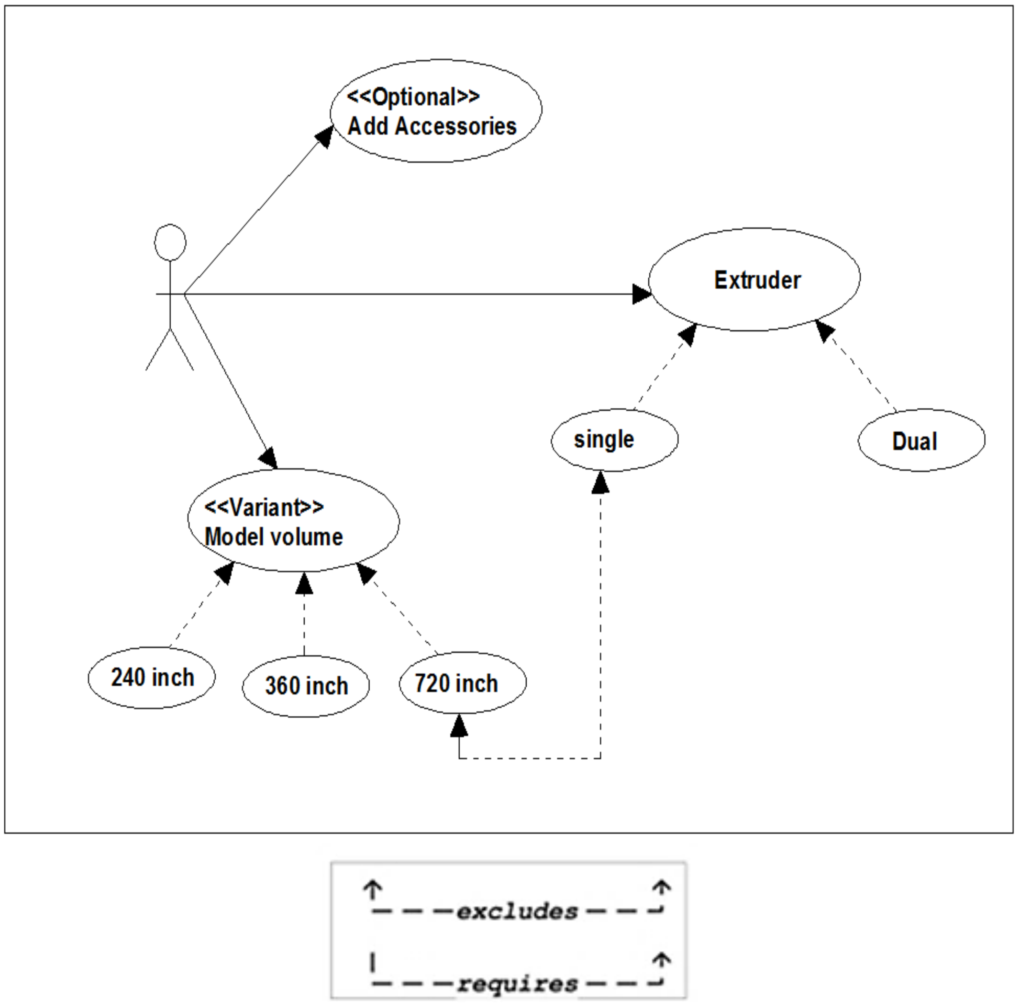


Figure 3. 7 Simplified UML Use case diagram for the 3D printer

Chapter 4

Feature Modeling in Depth

Software Product Lines are extensively expressed in terms of “features”. Explaining a feature as a term, is fundamental when it comes to variability modeling in software product lines, as various definitions are present in respect to various points of views (users, stakeholders, implementers). Thus, it is important to outline features and feature models clearly, as well as identify their terminologies and comprehensions accompanied by them. This chapter synthesizes a systematic review which aims to unwrap and identify features and feature modeling. In addition, during the course of this chapter, a state of the art of the present feature models is reflected thoroughly, stressing out their core traits and peculiarities. Successively, a holistic study is to be conducted for the evolution of feature models followed by their evaluation.

4.1. Feature Modeling: An Explication

Granting the frequent usage of the term “feature” in regards to the SPL, there is a scarcity of clear understanding of the term. The systematic literature review is carried out to firstly explore the varied definitions and potential utility of the term “feature” and “feature models” in SPL researches. This is tracked down by detailed exploration of feature modeling notations, approaches, specification to be shadowed by an evaluation of selected FM notations.

4.1.1. Review method

The systematic literature review is carried out to set an in depth study of feature modeling, starting with terming features, reaching a thorough investigation of FMs.

In accordance to a coordinated and systematic method, as per Keele (2007) guidelines and Kitchenham et al. (2009), 152 reference records dated from 1990 to 2015 in previous literature were identified as relevant with the research objectives. Those identified records are then subjected to thorough inspection, via systematic literature screening. Collected records are screened according to inclusion and exclusion criteria and the screened records are subjected to further filtering which leads finally to a selection of data records apt for synthesis in the literature review. Then narrative syntheses of various definitions are conducted, followed by the identification of their potential value for this examination.

4.1.2. Review goal

The systematic literature review is carried out to summarize the present literature in respect to feature modeling in SPL. Evidences from previous literature are pointed out to outline various definitions, explore and provide better understanding, and identify any possible gaps in order to suggest areas for further investigation. In addition, this review is carried out to have an in depth understanding of the evolution of the available feature modeling notations. Also, this review helps in providing a solid background to build on the following parts of the research.

4.1.3. Inclusion and exclusion criteria

The starting point for the adoption of the studies is the inclusion and exclusion criteria.

We review studies from selected sources (listed in subsection 0) according to the following inclusion criterion:

- IC1. Peer reviewed papers are included from year 1990 to 2015.

In this manner, papers from the dawn of FM until their most recent appearance are considered.

- IC2. Full studies including literature reviews related to our topic and research goals are included.
- IC3. Articles and papers in light of feature modeling notations in SPL.

These materials are precisely dedicated to our focus and abide to our review goals. The studies that were included in our review were directly related to our topic with clear defined structure founded on previous accurate studies or theoretical studies. However our exclusion criterion is based as follows:

- EC1. Sketchy and superficial studies that lacks in-depth and exhaustive studies.

Like so, trivial and inconsequential considerations are avoided and eliminated.

- EC2. Incomplete studies, that ranges within extended abstracts, tutorials or presentations.

These are disregarded as they provide curtailed and limited support as well as lack in depth.

- EC3. Papers and studies that are published before the year 1990 and after the year 2015.

As been rationalized in the inclusion criteria, the review takes an exclusive interest in the published papers from the emergence of FM till the most recent work.

- EC4. Articles outside the subject discussed.

Those articles are omitted as they don't fulfil our review goals and are irrelevant of the theme of study.

- EC5. Studies that presents subjectivity in which the author presents a subjective point of view or biased information.

In this way, preconceptions are avoided and integrity is maintained.

- EC6. Papers that doesn't relate to SPL specifically.

As per example, papers only relating to computer science or artificial intelligence are excluded, since they are not specific to our domain of study.

4.1.4. Data basis and selection criteria

According to Keele (2007), Primary data records are identified and screened in a manner that the records provides distinct and direct connection to feature modeling in SPL.

Subsequently, duplicates are removed and records are subjected to further refinements which lead to further exclusion of records with bias and records with insufficient information. Thus, the assessment provided reliable records proceeded by clear interpretation of outcome.

The following sources of information are used as data basis and sources:

- Designated Noticeable and prominent conferences concerned of features and feature modeling in SPL, which include:

  International Software Product Line Conference (SPLC), IEEE International Conference, , Requirements Engineering Conference (RE), International Conference on Software Reuse, International Software Product Line Conference, Conference in Computing Science, and International Conference on Software Engineering (ICSE).

- Various leading journals and reports related to feature and feature modeling in SPL, such as:

  IEEE Transactions on Software Engineering[1], ACM on Software Engineering[2], Journal of Systems and Software, Software Quality Journal, Journal of Theoretical and Applied Information Technology, International Journal of Advanced Manufacturing Technology, International Journal of Technology Management, Overload Journal 78 and Software Quality Journal

[1] http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=32
[2] http://tosem.acm.org/

- Workshops in the aforementioned prominence, namely:

International Workshop on Domain Engineering, International Workshop on Requirements Engineering for product lines, Workshop on Variability Modeling of Software-Intensive Systems, International Workshop on Satisfiability Modulo Theories, International Workshop on Requirements Reuse in System Family Engineering, International Workshop on Software Factories, Workshop on Software Variability Management for Product Derivation.

- Book Chapters and other significant sources are also considered in the review.
- References extracted from the preliminary journals are included in the synthesis in order to incorporate former data sources.

4.1.5. Data Hunt and assessment

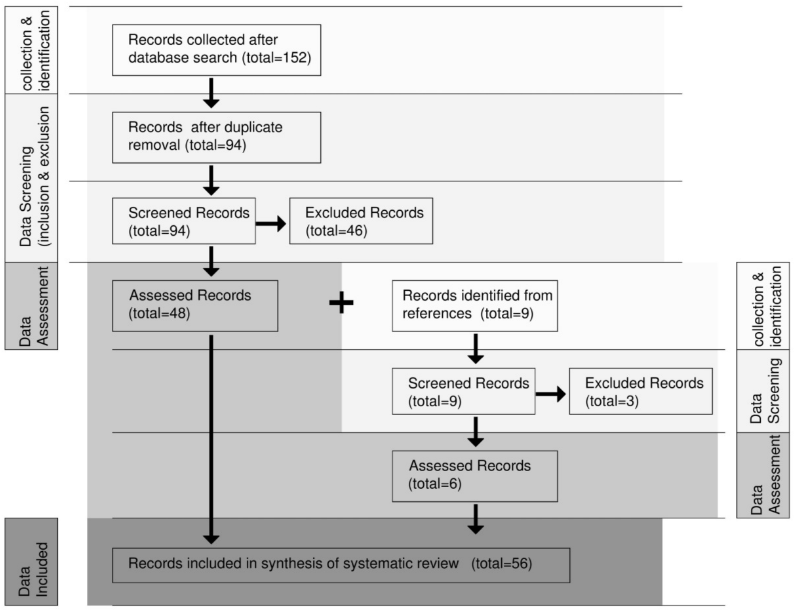
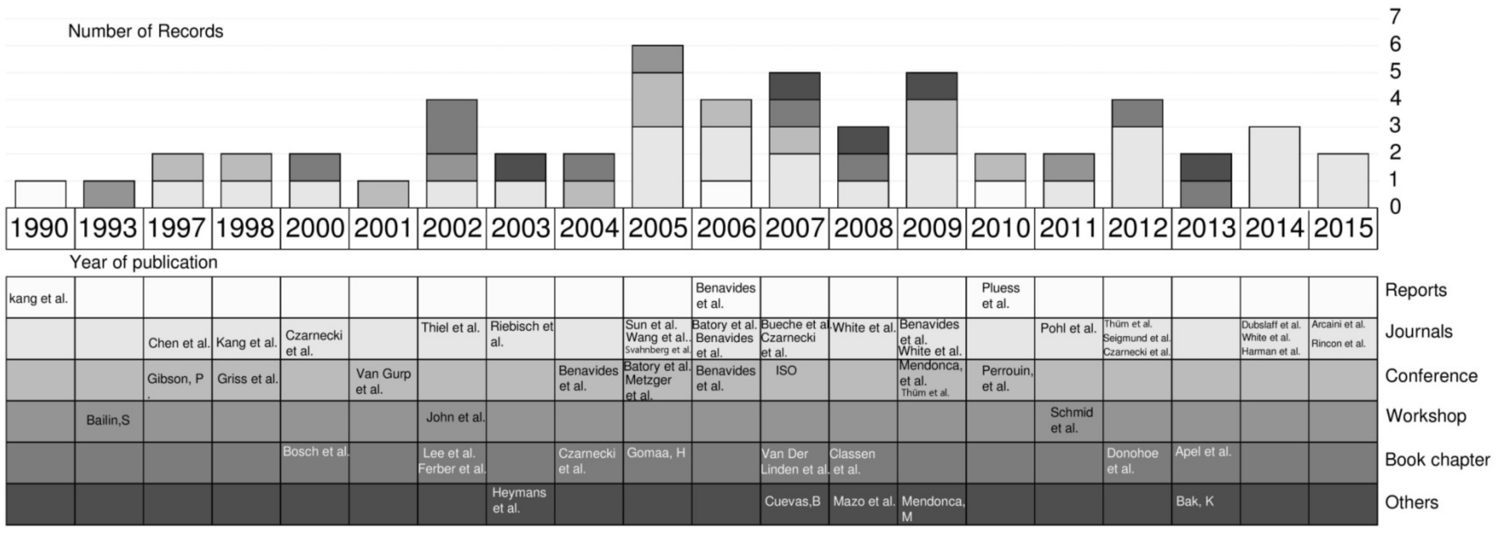

collection & identification
Records collected after database search (total=152)
Data Screening (inclusion & exclusion)
Records after duplicate removal (total=94)
Screened Records (total=94)
Excluded Records (total=46)
Data Assessment
Assessed Records (total=48)
+
Records identified from references (total=9)
collection & identification
Screened Records (total=9)
Excluded Records (total=3)
Data Screening
Assessed Records (total=6)
Data Assessment
Data Included
Records included in synthesis of systematic review (total=56)

Figure 4. 1 Review Data assessment

Figure 4. 1 illustrates the data hunt and further assessment of records, which are undertaken according to the following criteria:

- Data are firstly collected and identified from the aforementioned journals, conferences, book chapters, reports, workshops and other materials.

- The Data hunt is initiated on the basis of titles and search words (key words) related to feature modeling in SPL. The initial records identified formed 152 papers.
- Abstracts are then examined to figure out relevant sources and narrow down the search (thorough examination for some sources is undertaken when necessary). As a result, Duplicates are excluded.
- The final stage included examining the selected records judiciously to determine if they fit the purpose of the review.
- In consequence, irrelevant materials are eliminated to narrow down the search to 48 potential sources.
- References relevant to our search purpose (11 in total) are then identified and screened to be narrowed down to 6 sources to be added to the assessed records.
- The total number of records involved in the review formed 56 records at the end.



	1990	1993	1997	1998	2000	2001	2002	2003	2004
Reports	kang et al.								
Journals			Chen et al.	Kang et al.	Czarnecki et al.		Thiel et al.	Riebisch et al.	
Conference			Gibson, P.	Griss et al.		Van Gurp et al.			Benavides et al.
Workshop		Bailin,S					John et al.		
Book chapter					Bosch et al.		Lee et al. Ferber et al.		Czarnecki et al.
Others								Heymans et al.	

	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Reports		Benavides et al.				Pluess et al.					
Journals	Sun et al. Wang et al. Svahnberg et al.	Batory et al. Benavides et al.	Bueche et al. Czarnecki et al.	White et al.	Benavides et al. White et al.		Pohl et al.	Thüm et al. Seigmund et al. Czarnecki et al.		Dubslaff et al. White et al. Harman et al.	Arcaini et al. Rincon et al.
Conference	Batory et al. Metzger et al.	Benavides et al.	ISO		Mendonca, et al. Thüm et al.	Perrouin, et al.					
Workshop							Schmid et al.				
Book chapter	Gomaa, H		Van Der Linden et al.	Classen et al.				Donohoe et al.	Apel et al.		
Others			Cuevas,B	Mazo et al.	Mendonca, M				Bak, K		

Figure 4. 2 Data classification matrix

Figure 4. 2 shows a detailed classification matrix of the records used in the systematic literature review in accordance to the year of publications, ranging from 1990 to 2015, and the record type.

the 56 records includes 3 technical reports, 26 journal papers, 11 conference papers, 3 workshop papers, 8 book chapter and 5 from other materials.

4.1.6. Review Outcome

- **Terming "Feature"**

In regards to the term feature, 15 records of the 56 records used the term feature as an increment to function, 10 records referred to it as a specification in modeling in SPL, Whereas, 9 papers identified feature as a property, behavior or quality of software system. Table 4. 1 Feature's definition in literature and Figure 4. 3 Feature's definition classification illustrate the findings.

Feature definition in literature		
Increment to function	Specification	Behavior or Quality of Software System
(Cuevas, 2007)	(Benavides et al., 2007)	(Kang et al., 1990)
(Benavides, Ruiz-Cortés, Corchuelo, & Martín-Díaz, 2004)	(Benavides, Cortés, et al., 2006)	(Lee et al., 2002)
(Kang et al., 1990)	(Czarnecki & Eisenecker, 2000)	(Griss et al., 1998)
(Benavides, Cortés, Trinidad, & Segura, 2006)	(Bailin, 1993)	(Kang et al., 1998)
(Batory, Benavides, & Ruiz-Cortes, 2006)	(Svahnberg et al., 2005)	(John & Muthig, 2002)
(Dubslaff, Klüppeholz, & Baier, 2014)	(K. Chen, Zhang, Zhao, & Mei, 2005)	(Wang, Li, Sun, Zhang, & Pan, 2005)
(White et al., 2009)	(Batory, 2005)	(Bosch et al., 2001)
(Metzger et al., 2007)	(Classen, Heymans, & Schobbens, 2008)	(Rincón et al., 2015)
(White et al., 2014)	(Apel et al., 2013b)	(Van Gurp et al., 2001)
(White, Schmidt, Benavides, Trinidad, & Ruiz-Cortés, 2008)	(Beuche & Dalgarno, 2007)	
(Siegmund et al., 2012)		
(Gomaa, 2005)		
(Batory, 2005)		
(Zave & Jackson, 1997)		
(Gibson, 1997)		

Table 4. 1 Feature's definition in literature

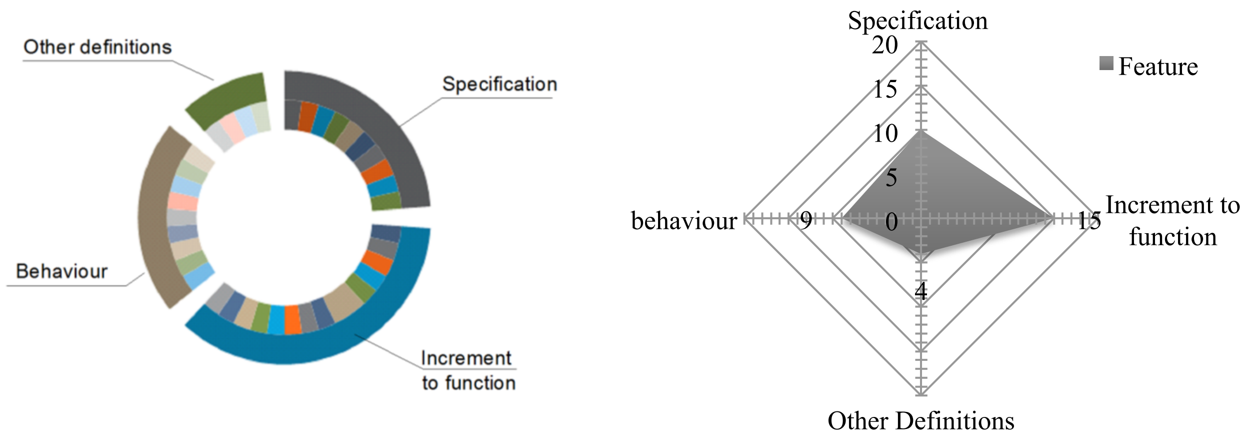


Figure 4. 3 Feature's definition classification

- **A selection of Notable definitions**

It is important to point out notable terminologies for feature in SPL.
As per the systematic literature review, features are approached in accordance to three main definitions which are;
features as an increment to function, features as specification for modeling in SPL, and features as a behaviour or quality in software systems.
Other definitions identified in literature are also worthy to point out.

Starting with feature as increment to function, Kang et al. (1990) first defined Features as distinguishable functional abstractions that must be incremented, established, delivered, and maintained. Moreover, feature is defined as an increment in product functionality in software systems according to Cuevas (2007).
Siegmund et al. (2012) , Zave and Jackson (1997) and Batory (2005) stresses out the same definition of features as an increment to function which satisfies and fulfills functional requirements.

Features are also referred as a specification, as per Benavides et al. (2007) software systems are specified in terms of features. In addition, according to Kang et al. (1990), a product of a product line is specified by a valid feature selection.

Moreover, features are seen to be unique user-visible aspect, quality, behavior or characteristic of software system (Kang et al., 1990), Whereas Bosch et al. (2001), sees feature as "a logical unit of behavior that is specified by a set of functional and quality requirements".

- **Terming "Feature modeling"**

Mainly, features in software product lines are expressed in feature models, which bring out the importance of clearly defining feature models.
This part introduces various definitions and perceptions of feature models extracted from previous literature.

	Feature models definitions	
Info representation	Commonalities and variability tool	Specification member in product lines
(Benavides et al., 2009)	(Benavides et al., 2007)	(Mendonca, Wąsowski, et al., 2009)
(Benavides et al., 2004)	(Czarnecki & Eisenecker, 2000)	(Benavides, Cortés, et al., 2006)
(Mendonca, Wąsowski, & Czarnecki, 2009)	(Batory et al., 2006)	(Batory, 2005)
(Benavides, Cortés, et al., 2006)	(Mendonca, Wąsowski, et al., 2009)	(Czarnecki & Eisenecker, 2000)
(Batory, 2005)	(Lee et al., 2002)	(Lee et al., 2002)
(Benavides, Segura, Trinidad, & Ruiz-Cortés, 2006a)	(Kang et al., 1998)	(Thüm et al., 2009)
(Mendonca, Branco, & Cowan, 2009)	(White et al., 2009)	(Thiel & Hein, 2002)
(Benavides, Segura, Trinidad, & Ruiz-Cortés, 2006b)	(White et al., 2014)	(Riebisch, 2003)
(John & Muthig, 2002)	(Mendonça, 2009)	(Pleuss, Botterweck, Dhungana, Polzer, & Kowalewski, 2010)
(Mazo & Salinesi, 2008)	(Czarnecki & Wasowski, 2007)	(Arcaini et al., 2015)
(Czarnecki, Grünbacher, Rabiser, Schmid, & Wąsowski, 2012)	(R. Pohl, Lauenroth, & Pohl, 2011)	
(Rincón et al., 2015)	(Thüm et al., 2009)	
(Arcaini et al., 2015)	(Czarnecki, Helsen, & Eisenecker, 2004)	
(Heymans et al., 2008)	(Sun, Zhang, Fang, & Wang, 2005)	
(Harman et al., 2014)	(Wang et al., 2005)	
(Beuche & Dalgarno, 2007)	(Mazo & Salinesi, 2008)	
(Bak, 2013)	(Beuche & Dalgarno, 2007)	

Table 4. 2 Feature model's definition in literature

Figure 4. 4 and Table 4. 2 Feature model's definition in literature point out various definitions and perceptions of feature modeling in SPL. According to the systematic literature review, Feature modeling is regarded as information presentation for SPL in 17 records of the 56 records assessed. Whereas, the term feature modeling is referred as a commonality and variability identification technique in 19 records. However, FMs are also referred as specification member in SPL according to 10 records.

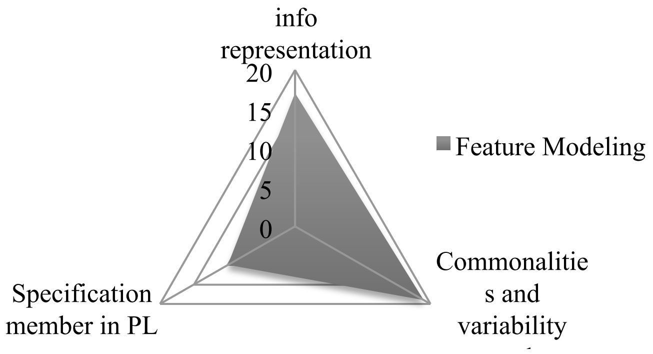


Figure 4. 4 Feature modeling's definition classification

- **A selection of Notable definitions**

A feature model can be used to represent the similarities and differences within a family of software systems (Mendonca, Branco, et al., 2009). The combination of features represents all variability, thus forms FM (John & Muthig, 2002).
Czarnecki and Wasowski (2007), however, defines Feature modeling as a notation and an approach for modeling commonality and variability in product families.
Furthermore, Lee at al. (2002) adopts a comprehensive approach in which he defines FM as the activity of identifying externally visible characteristics of products in a domain and organizing them into a model called a feature model.

- **Concluding Remarks**

It is apparent that clear common definitions for feature and feature modeling have not been agreed on, in which features are considered as function, property and specification (Classen et al., 2008). However, Tracking down a consensus on their definition and clarifying the terminologies of "Feature" and "feature modeling" is an essential step for more thoroughly investigating of its mechanisms. Although it seems

an unfeasible task where biases and no uniformity in defining the terms exist, Maturing a clearer understanding of the feature and feature modeling gives insights essential to shaping the background for new approach to feature modeling to be discussed in the following part of dissertation.
In this thesis, features are considered as properties, while FM is a mean to express these features, their dependencies and constraints as well as commonalities and variabilities.

4.2. Feature Modeling Exploration

Feature modeling weighs in greatly in SPL, in which it aims to present commonalities and variabilities amongst features, as well as highlight their dependencies and constraints (Czarnecki, Helsen, & Eisenecker, 2005; Kang et al., 1998; Riebisch, 2003). This section helps to digest FM systems, principles and types.
As explicated, features display information of the SPL in the form of features and all their relationships (Benavides et al., 2009). Represented by feature diagrams, the set of features are arranged as follows:

- Hierarchically representing relationships between parent and child feature (also defined as sub features).
- Crosstree constraints which shows inclusions and exclusions, feature dependencies and interdependencies, requirements and alternative roots.

The Relations between a parent and the child features are considered as follows (Batory, 2005) (see Figure 4. 5):

- And; in which all child features are to be chosen
- Alternative; in which one alternate child feature can be chosen,
- Or; in which one or more child feature can be chosen,
- Mandatory; in which the selection of specific features are compulsory,
- Optional; in which the selection of features are optional.

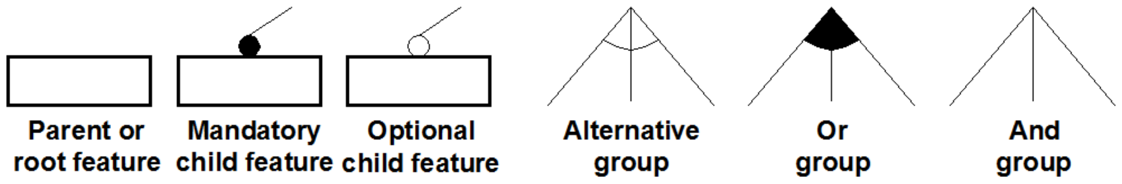


Figure 4. 5 Feature diagram graphical notations

4.2.1. Feature modeling notations

Notations of FMs are addressed as follow:

First we consider Basic FM, a primitive format of feature model.

4.2.1.1. Basic

In the basic notation, the FM is mostly a tree diagram which is diagrammatically presented by means of nodes. These nodes cast out the relationship between parent features and sub features and forms out the joints and connections between them.

Those relationships in the basic notation are considered to be primitive relations to be explained as below (Batory, 2005; Benavides et al., 2009; Lee et al., 2002):

- **Mandatory relationships**

In Mandatory relationships, the sub features is necessary to be included when the parent is included and in no means should the sub feature be excluded when the parent is included.

This mandatory relationship is symbolized by a line connecting the parent feature with the sub feature with a black filled circle above the sub feature connection. Figure 4. 6 illustrates a Mandatory relationship extruded from the exemplary 3D printer feature model. In Figure 4. 6, we can point out the 3D printer mandatory comes with a body, which consequently mandatory consists of an extruder and a case.

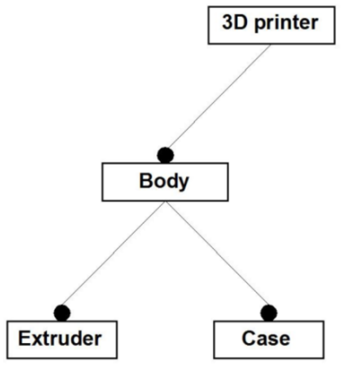


Figure 4. 6 Mandatory relationship

- **The optional relationship**

Optional relationship is the relationship between parent features and sub features, in which the sub features inclusion is optional, hence the existence of this sub feature is conditional.

The optional relationship is symbolized by a line connecting the parent feature with the sub feature with an empty circle on top of the sub feature connection. Figure 4. 7 exemplifies the optional relationship extruded from the exemplary 3d printer FM.

The figure below demonstrates the 3d printer optional relationship in terms of including accessories or excluding them. However on the other hand the inclusion of the body, model, connectivity and software are a must.

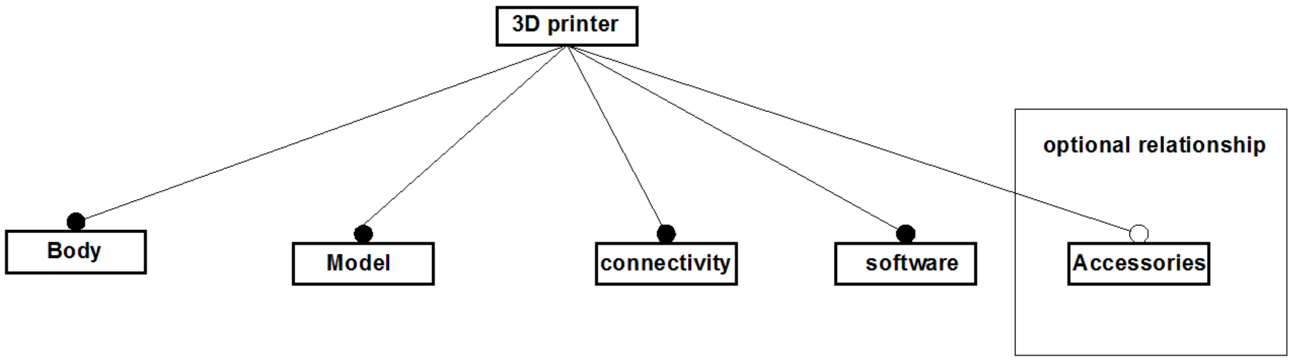


Figure 4. 7 Optional relationship

- **The or-relationship**

The Or relationship presents the relationship between the parent and the sub feature, in which minimum of one sub feature is to be selected in terms of the inclusion of the parent feature. The Or relationship is symbolized by a black filled arc connecting the array of lines connecting the parent feature and the sub features.

In Figure 4. 8, the Or relationship of a parent feature and its sub feature is demonstrated, extracted from the 3d printer feature model example. In this case the options of the software derive from choosing at least one feature (such as Sketchup) till choosing all four features.

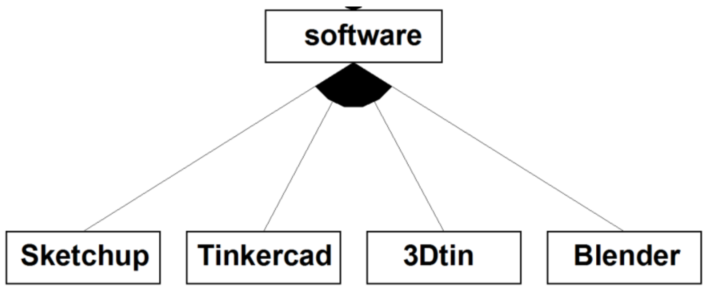


Figure 4. 8 Or relationship

- **The alternative relationship**

The Alternative relationship portrays the association of the parent and the sub features, in which one and only one option of the sub features is to be selected, when the parent feature is to be included, thus presents an alternate choice of sub features. The alternative relationship, in other words is an Exclusive Or relationship XOR, symbolized by an arc connecting the array of lines connecting the parent feature and the sub features (see Figure 4. 9).

The following figure demonstrates the Exclusive Or relationship of a parent feature and its sub feature, also extracted from the 3D printer FM example. In terms of the Exclusive Or relationship, The volume alternates from 240 inch, 360 inch and 720 inch, in which only one volume is to be selected.

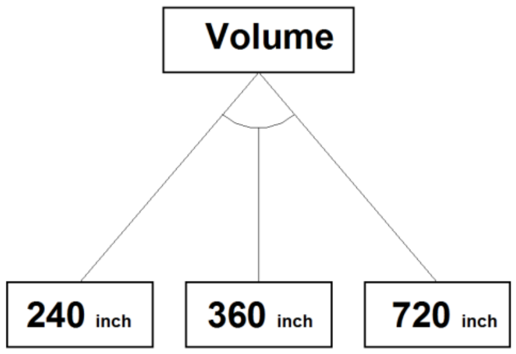


Figure 4. 9 Alternative relationship

In addition to the relationships explained above, basic feature models are accompanied by constraints relationships:

- **Requires constraint**

When one feature is to be included, and this feature must be accompanied by another feature, then "requires" constraint is present.

- **Excludes constraint**

When one feature is included in the diagram, and this feature requires the elimination of the presence of another feature, then the exclusion is present, and hence "excludes" constraint which resembles the incompatibility of 2 features.

4.2.1.2. Cardinal notation

The urge of the completeness in concept and the need of practical application (Czarnecki & Eisenecker, 2000) formed a motivation for the extension of feature modeling notation, and thus the existence of the cardinality based FM. This notation is defined by the feature and group cardinalities as below (Czarnecki, 2005):

- **Feature Cardinality**

The extension of the feature model from basic to cardinal inserted some information and labels to the relationship between parent feature and sub feature.

It inherited a UML-like multiplicity having the form of intervals [n, m], where an upper and lower bounds are presented. This is mostly expressed by means of arbitrary numbers. By these means, we can present, as well as limit the number of sub features that are included, when the parent feature is included.

As an example, a mandatory relationship is expressed as [1, 1] meaning the sub feature is to be compulsory selected. Whereas on the other hand, [0, 1] provides the option of choosing from the upper to the lower bound, and thus resembling the optional relationship.

- **Group Cardinality**

The group cardinalities is adopted when considering the Or relation and the Exclusive Or relationship between parent feature and sub features. The interval in this case is denoted by {n…m} having also n,m as limiting bounds of upper and lower limit of the number of sub features to be selected when the parent feature is included.

When considering the Or relationship in which one and more sub features can be selected, the interval is {1..N} in which N presents the number of sub features. However, when considering the alternative relationship, in which only feature can be selected the interval in this case is {1..1} showing the route of only selecting one sub feature.

4.2.1.3. *Extended Feature model*

Besides the basic and the cardinality feature modeling notation, additional information that distinguishes features traits are necessary to describe features (Batory, 2005; Czarnecki & Kim, 2005; Kang et al., 1998). Originating from this point, the extended feature modeling notation took place to add up those additional information known as feature attributes. The extended feature model primitive establishment was in present in FODA (Kang et al., 1998) in which additional information is included in FODA feature model.

There have been no approved definition of what feature attribute is to include, however, fundamentally speaking, the feature attribute must consist of domain, name and value and it can add up any other trait distinguishing the feature such as cost, size, speed…
Figure 4. 10 illustrates an example of extended feature model presented by Benavides et al. (Benavides et al.).

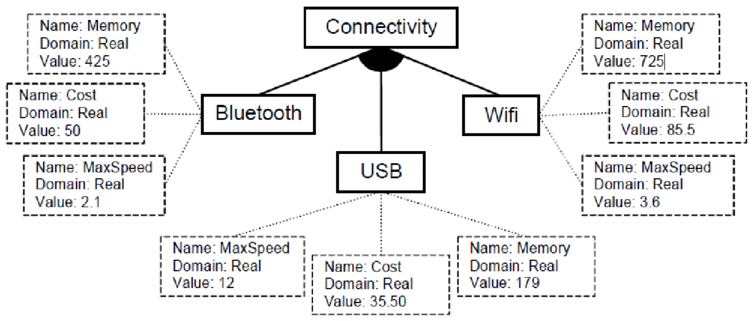


Figure 4. 10 Extended feature model notation (Benavides et al., 2009)

4.3. Evolution and evaluation of FMs

The rise of the FMs relatively is associated to the domain analysis in SPL. It firstly emerged with the emergence of SPL systems in telecommunications and later on extended to reach a vast array of fields (Pleuss et al., 2010).

FMs presented means of communication and demonstration of information about the products in the system, which reveals requirements accompanied in the product, hence providing a smoother and more transparent system for developers, customers and stakeholders, as well as an ease in design. Moreover, feature models are used to facilitate the selection of features, ease configuration and automate the formation of the SPL. In addition, Feature models helps in capturing commonalities and variabilities.

Kang (1990) is considered to be the initiating author who took the lead in feature modeling. Feature modeling was firstly evolved from the feature oriented domain analysis 'FODA' (Kang et al., 1990). The evolution formed different features models developed by various authors, as to adapt to the evolving uses or to fix present glitches in previous feature models.

A timeline of feature models is drawn to show their evolution starting from 'FODA' developed by Kang et al. in 1990 which is followed by Jacobson et al. attempt in 1997. Later on, Kang developed the original feature model 'FODA' to 'FORM' (Feature oriented reuse method) in 1998 which evolved to 'FOPLE' in 2002. Moreover in 1998, Griss et al. developed the 'FeatuRSEB' feature model that was later evolved by Van Gurp et al. in 2001 and Eriksson et al. in 2005. A huge leap in feature modeling evolution is taken by Czarnecki et al. in 2000, in which generative programming feature model 'GP' was developed which branched out and opened new opportunity to evolve Riebisch et al. feature model in 2002, and GP extended feature model developed by Czarnecki et al. in 2002. The GP extended Feature model was evolved by Czarnecki et al. in 2004 to form Cardinality based feature model. It's also important to pinpoint Benavides et al. feature model in 2005. However, the list isn't limited by these feature models, various feature model were developed, but in this section we shed the light on the most significant feature models that contributed in SPL systems in order to evaluate and learn from them.
Table 4. 3 shows the timeline of various feature models present in SPL systems.

Feature Model	YEAR	Feature Model	YEAR
Kang et al. FODA	1990	Czarnecki et al. GP extended	2002
Jacobson et al.	1997	Riebisch et al. FORE	2002
Kang et al. FORM	1998	Gomaa et al.	2004
Griss et al. FeatuRSEB	1998	Czarnecki et al. CBFM	2005
Czarnecki et al. GP	2000	Moon et al.	2005
Hein et al.	2000	Eriksson et al. PLUSS	2005
Van Gurp et al.	2001	Benavides et al.	2005
Kang et al. FOPLE	2002		

Table 4. 3 Feature modeling notations timeline

Figure 4. 11 clarifies the evolution of feature model, in which FODA, the head of the pyramid, presents the mere beginning of feature modeling attempts and is tracked down by several extensions not limited to those presented.

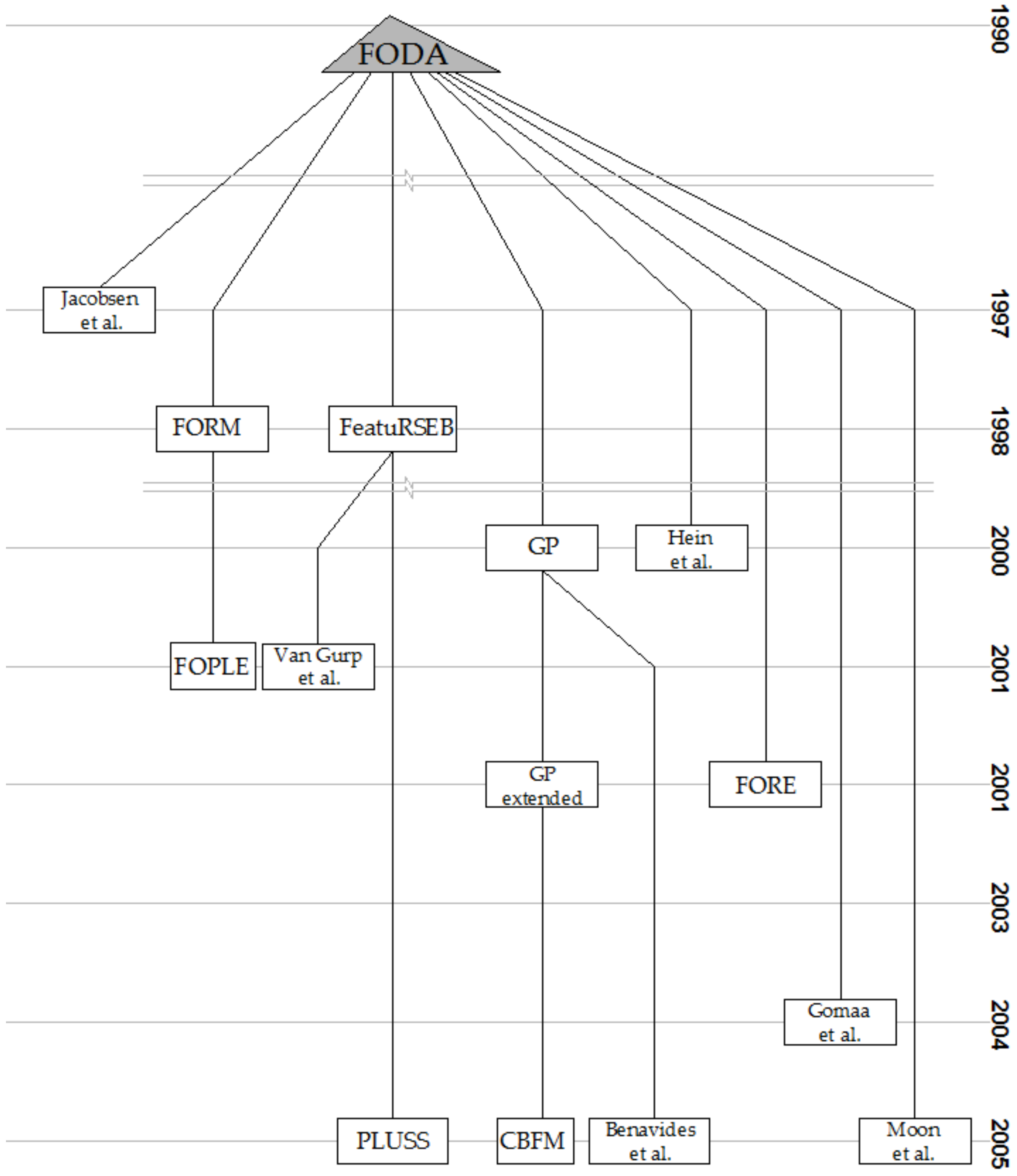
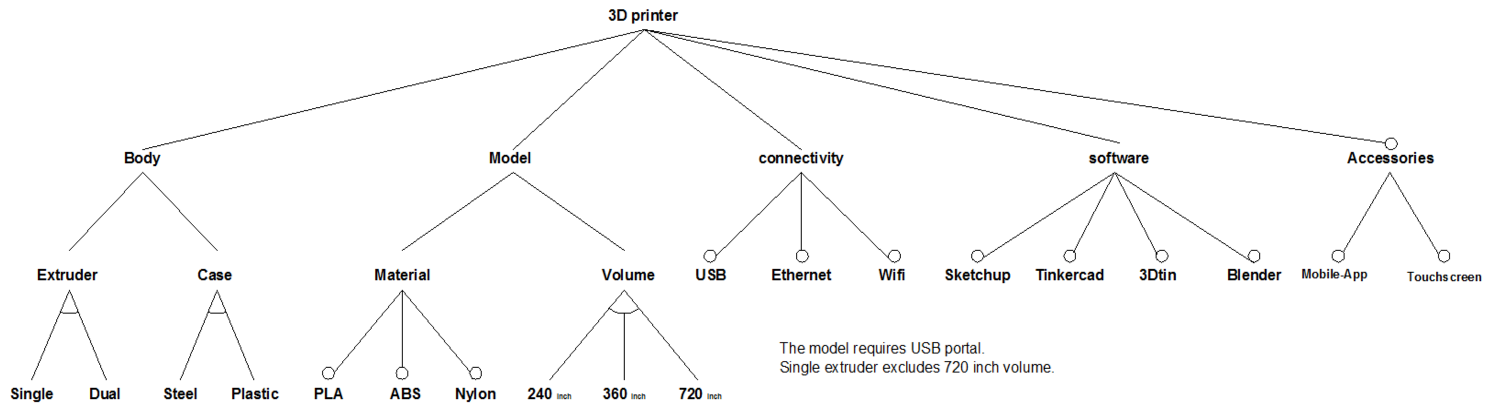


Figure 4. 11 Feature models evolution

4.3.1. Feature Oriented Domain Analysis

FODA, the Feature Oriented Domain Analysis method (Kang et al., 1990), is the first feature diagram notation to be introduced in SPL systems (L. Chen et al., 2009). FODA was represented by a tree graph illustrating commonalities and variabilities in the feature model. FODA provided an ease of use and a method of communication between customers, developers and stakeholders.

The FODA diagram notations, compositions, relationships and constraints are illustrated in the Figure 4. 12 and Table 4. 4 below and explained as follows:



The model requires USB portal.
Single extruder excludes 720 inch volume.

Figure 4. 12 FODA modeling notation of 3D printer

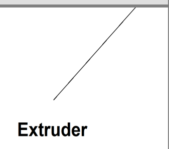
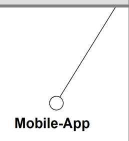
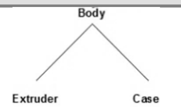
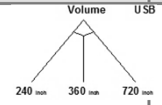
Feature Relationships						Constraints	
Parent feature	Mandatory	optional	And	Or	Xor	Textual	graphical
3D printer				✗		✓	✗

Table 4. 4 FODA notation specifications

4.3.1.1. Features

- The main feature, also known as root feature or concept; is a mandatory feature that represents the whole system. According to the example above, the root feature is 3D printer.
- Mandatory features exists by default; there is no graphical notation or expression specific to mandatory features (example Body, Model, Material ...)
- Optional features are represented by an empty circle above the connected sub feature, as per example accessories.

4.3.1.2. Relationships

- And relationship, implying that all subfeatures are to be selected in the system, the "And" relationship doesn't have any specific expression. It shows the mandatory features to be included in the system.
- Xor relationship presents the relationship in which one and only one sub features is to be selected, when the parent feature is to be included. The xor relationship is expressed by an arc connecting the array of lines connecting the parent feature and the sub features.

  In Figure 4. 12 FODA modeling notation of 3D printer, the Xor relation is demonstrated, in which only one volume can be selected from the alternatives: 240 inch, 360 inch and 720 inch.

4.3.1.3. Constraints

FODA model lacks any graphical representation of constraints, however textual constraints are present describing requires and excludes constraints.

4.3.2. Feature-Oriented Reuse Method

FORM, the Feature-Oriented Reuse Method, is an add on of FODA developed by Kang et al. (Kang et al.). It emerged as a necessity to widen the scope of feature modeling in terms of domain analysis, and implementations.

FORM has managed to build upon the previous feature diagram by changing and adding up elements, such as;

- The graphical representation can be directed acyclic graph, as well as tree representation.
- Implemented "By" relationship, along with the And and Xor relationship.
- Generalization and specialization also exists in FORM model.
- The graphical representation of the feature and sub features themselves are within boxes (see Figure 4. 13 and Table 4. 5).

However, FORM still lacks Or- relationship and graphical constraints.

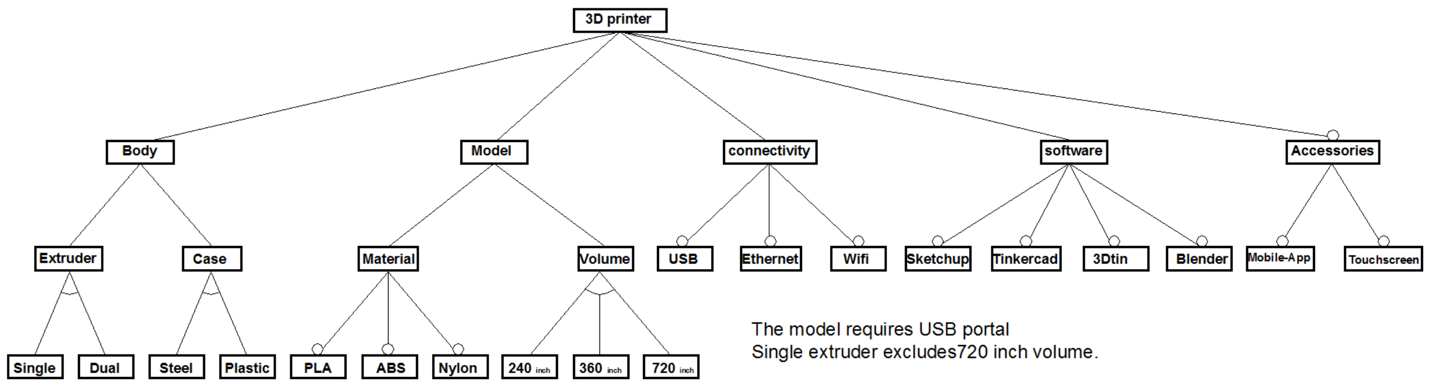


Figure 4. 13 FORM modeling notation of 3D printer

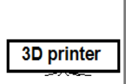
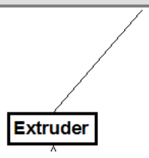
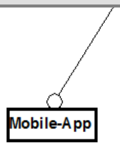
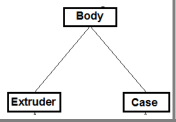
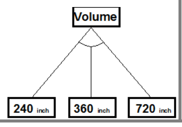
Feature Relationships						Constraints	
Parent feature	Mandatory	optional	And	Or	Xor	Textual	graphical
				✗		✓	✗

Table 4. 5 FORM notation specifications

4.3.3. FeatuRSEB

FeatuRSEB (Griss et al., 1998) is derived from the feature oriented domain analysis method and the RSEB method, which depends on use cases as well as object models. FeatuRSEB sheds the light on variation points and variants in the graphical representation.

FeatuRSEB is distinguished as follows;

- The graphical representation is directed acyclic graph DAG.
- The And relationship is similar to that described in FODA.
- The Xor and or relationship are represented by means of parent features, known as variation points and sub features, known as variants.
  The Xor relationship is resembled by an empty diamond shape connecting the variation points and the variants. Whereas, the Or relationship is presented by a black diamond connecting the variation points and the variants (see Table 4. 6 and Figure 4. 14).
- Textual constraints, along with graphical constraints, are present which are "requires" and "excludes".
- FeatuRSEB provides binding time; which are reuse-time and use-time binding.

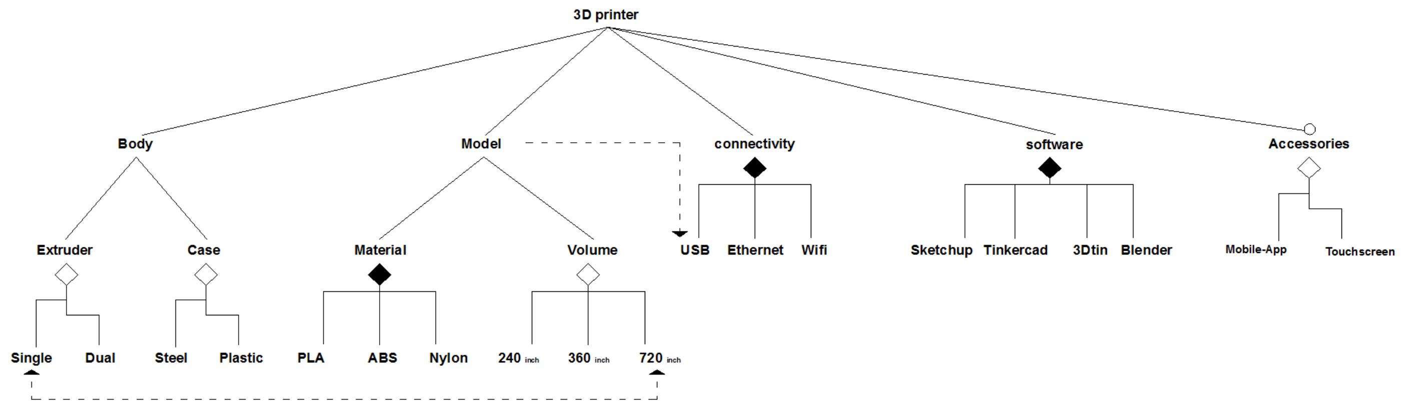


Figure 4. 14 FeatuRSEB modeling notation of 3D printer

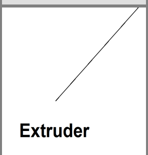
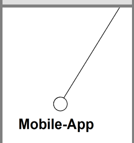
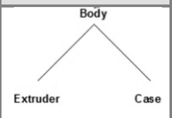
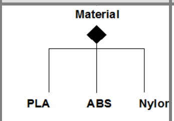
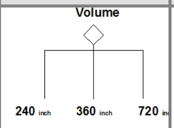
Feature Relationships						**Constraints**	
Parent feature	Mandatory	optional	And	Or	Xor	Textual	graphical
3D printer						✓	✓

Table 4. 6 FeatuRSEB notation specifications

4.3.4. Generative programming

Generative programming GP was developed by Czarnecki et al. in 2000, deriving from FODA, and creating a lead in terms of software automation programing.

The feature diagram is known as Generative Programming Feature Trees, GPFT.

As being derived from Kang et al feature model (Kang et al.), the generative programming feature diagram adds up on FODA's graphical representation.

- Or relationship is added, which presents the relationship in which at least one or more subfeatures is to be selected, when the parent feature is to be included. The Or relationship is expressed by a black filled arc connecting the array of lines connecting the parent feature and the sub features (see Figure 4. 15 and Table 4. 7).
- Mandatory features exist in the condition of the inclusion of its parent feature. Mandatory features are represented by a black circle above the connected sub feature.

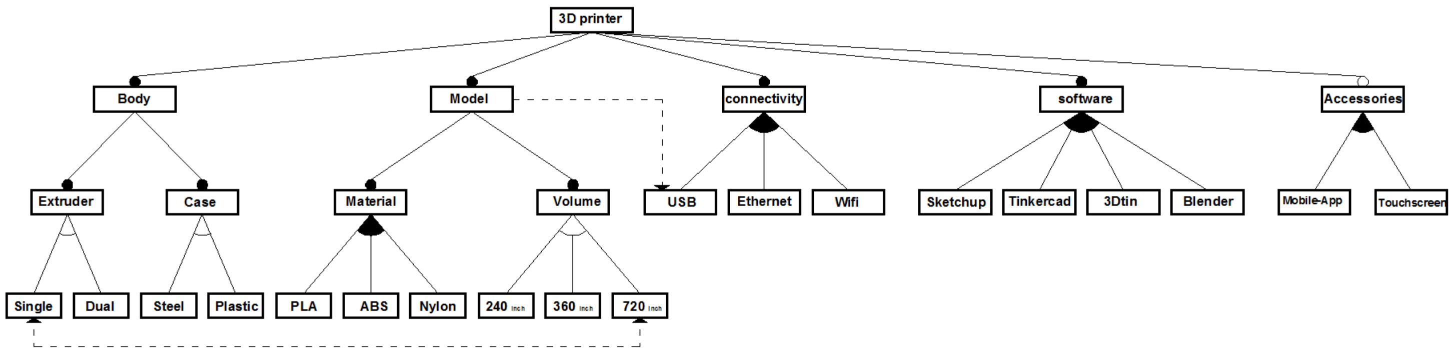


Figure 4. 15 GP modeling notation of 3D printer

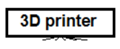
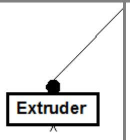
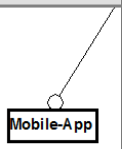
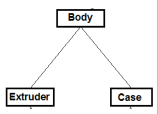
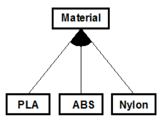
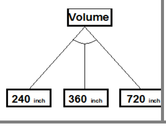
Feature Relationships						Constraints	
Parent feature	Mandatory	optional	And	Or	Xor	Textual	graphical
						✗	✓

Table 4. 7 GP notation specifications

4.3.5. Van Gurp and Bosch Feature Model

Van Gurp and Bosch developed a feature model as an extension of FeatuRSEB, in which it evolves to add up more specifications and characteristics. The feature diagram is known as van Gurp and Bosch Feature Diagrams VBFD (Van Gurp et al., 2001).

- The Van Gurp and Bosch Feature diagram varies from FeatuRSEB in terms of the Or and Xor notation, in which the Xor relationship is resembled by an empty triangle (instead of diamond) connecting the variation points and the variants. Whereas, the Or relationship is presented by a black filled triangle connecting the variation points and the variants (see Table 4. 8 and Figure 4. 16).
- VBFD provides enhanced dealing with binding times, as per annotating relationships between features.
- Features (the parent features and all subfeatures) are represented in a box, Whereas, External features are boxed in dashed boxes.

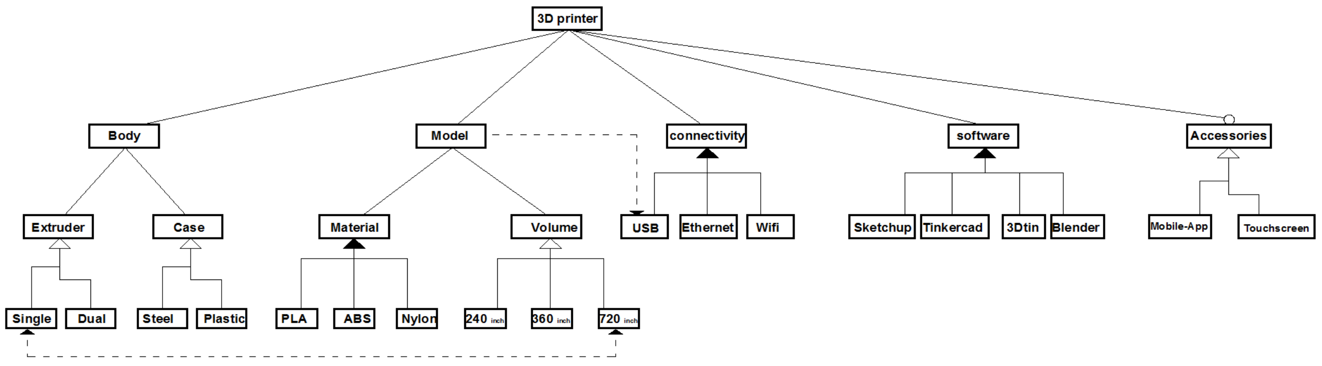


Figure 4. 16 Van Gurp and Bosch feature modeling notation of 3D printer

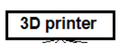
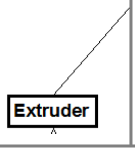
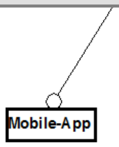
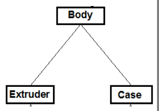
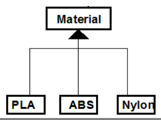
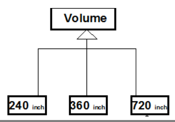
Feature Relationships						Constraints	
Parent feature	Mandatory	optional	And	Or	Xor	Textual	graphical
						✗	✓

Table 4. 8 VBFM notation specifications

4.4. Drawbacks of Feature Modeling

Although Numerous Feature modeling notation exists in SPL systems, those current FM notations are accompanied with shortcomings and glitches, limiting its efficiency and performance. In this chapter, an assessment of selected FM notations is undertaken to pinpoint the current problems in them.

The assessment takes place according to the criteria as set below.

4.4.1. Assessment criteria

When evaluating feature modeling notations glitches; the following criteria is to be considered.

FM notations are appraised on the basis of 5 key criterions which are: comprehensiveness, visual presentation suitability, traceability, scalability and articulacy.

The key criterions are summarized as follows:

4.4.1.1. Comprehensiveness

This criterion is used to assess the completeness and inclusiveness of the FM notations. It studies whether these notations fulfill a whole role in describing the features, their relationships, dependencies, and constraints.

This key criterion assessments is carried out by determining the following:

- The presence and comprehensiveness in terms of adequate feature relationships and constraints; which are but not limited to:
  - Relationships: OR, AND, optional, alternative and cardinalities.
  - Constraints: requires, excludes and/or textual constraints.
- Confusions and complications that accompanies the feature modeling notations in terms of inconsistencies and contradictions between constraints and relationships.

4.4.1.2. Visual presentation suitability

This key criterion takes into consideration the readability of the feature modeling notations. It studies the visual representation in terms of comprehensibility and adequacy.

This key criterion considerations are as follows:

- Completeness and entirety in information present in the FM notation.
- The graphical demonstration of interactions of features, in terms of the presence and absence of adequate visual presentation of constraints, relationships, variations and variation points...
- Readability and the ability to visually follow up, in which the FM graphical presentation is assessed in terms of complexity and sophistication.

4.4.1.3. Traceability

Traceability is an essential criterion which determines to what extent a feature model notation can be feasible and adaptive to changes. It takes into consideration the practicality and reliability of the feature modeling notation.

This key criterion concerns are as follows:

- The ability to evolve over time, integrate and adapt to changes.
- Data consistency and reliability in the feature modeling notations.
- The manageability of linking between features and requirements.

4.4.1.4. Scalability

The scalability of the feature modeling notation deals with the capability of this notation to handle large scale systems, as per dealing with complexity and large number of features, with taking into consideration all dependencies and constraints and satisfying the requirements.

This key criterion considerations are as follows:

- Determining to what extent the feature model notation can serve in large scale systems.

- Determining whether the feature model notation requires a support tool to handle scalability problems.

4.4.1.5. *Articulacy*

Uncertainties and ambiguities are key shortcomings in feature modeling notations. This criterion takes into consideration the articulacy and clarity in the feature modeling notations. It deals with glitches in term of misconstructions and misunderstandings.

This criterion addresses the following points:

- Ambiguity in interpreting and presenting relationships such as errors, double meanings, nonfunctional features and redundancies.

4.4.2. Appraisal and results

FODA, FORM, FeatuRSEB, GP, and VBFD Notations explained and demonstrated previously in section 4.3, are assessed according to the assessment criteria set above to determine their glitches and evaluate them.

The appraisal and results are summarized as follows:

4.4.2.1. *Comprehensiveness*

It is crucial to present a complete and inclusive interaction of features in the feature modeling notations. Consequently, as per the necessity to present the dependencies between features, their relationships and constraints, the lack of full presence of these interactions notations leads to shortcomings in the feature modeling approach, as it limits the interaction between features and leads to detrimental side effects. As features tends to be dependent and correlated to other features, any modifications on a sole feature, as per selection, deselection, addition or deletion, consequently has a direct effect on other related features. Thus, full comprehensiveness in the feature expressions as well as their interactions is crucial (Gibson, 1997).

When analyzing comprehensiveness of the selected features modeling notations, the findings are listed as follows:

- FODA, to begin with, presents primitive attempt to model feature interactions. Being a primitive feature modeling notation, FODA has shortage in modeling and presenting relationships and constraints as well as confusion. FODA modeling notation proved to lack completeness in concept associated with multiple glitches and gaps in its representation. According to Figure 4. 12 and Table 4. 4 FODA notation specifications, which illustrated the FODA feature model of a 3D printer, FODA lacks graphical notation of the Or relationship and the graphical constraints. The problematic concern is that information aren't well expressed in these structures, with a chance to overlook dependencies or create confined presentation with lots of lost information, in which relationships, dependencies and constraints are not expressed thoroughly.
- FORM was brought up to advance FODA approach (Kang et al., 1998). Similar to its predecessor FODA (Lee, Kang, Chae, & Choi, 2000), FORM has similar glitches in terms of comprehensiveness. Although FORM surpasses FODA by its encompassing to By relationship, in addition to the And and Xor relationships (see Figure 4. 13 and Table 4. 5), where one feature is implemented by means of another feature. Moreover, FORM puts in the generalization and the specialization relationship.
- FeatuRSEB, GP and VBFD adds up the graphical constraints, in which FeatuRSEB presents both textual and graphical "requires" and "excludes" constraints (see Table 4. 6) whereas GP and VBFD have only graphical constraints (see Table 4. 7 and Table 4. 8). Moreover, FeatuRSEB, GP and VBFD adds the Or, whereas only GP presents the mandatory relationship.

Having that said, the existence of the constraints and other mentioned relationships adds up the comprehensiveness in the model and promotes clearer understanding of the feature dependencies relationships, and less confusion in terms of inconsistencies and contradictions among constraints and relationships. However, the incomprehensibility and obscurity are still present between compositions and requires relations, and excludes and alternative. Thus, there's a need to elaborate and add up other dependencies expressions to avoid complications and confusions.

This implies that the considered feature modeling notations have glitches in terms of completeness and inclusiveness, in which none are completely comprehensive and expressive with a capability to achieve an unabridged role in providing clear expressiveness of the features, their relationships, dependencies, and constraints.

4.4.2.2. Visual presentation suitability

When talking about visual presentation suitability, all the modeling approaches assessed are graphically visualized and presented, knowingly it is crucial to spot the light on the variables in the model, as well as have a readable construct.

However the type of visualization varies from one model to another, thus the readability and the visual presentation suitability varies accordingly.

- As noted, FODA obeys a basic tree presentation which tends to be less inclusive and more problematic to create (Berg & Muthig, 2005). On the other hand, FORM and FeatuRSEB notations are Directed acrylic graph (Batory, 2005; Sun et al., 2005). Whereas, GP and VBFM notations are of tree structure which is more clear(Czarnecki & Helsen, 2003; Van Gurp et al., 2001).
- FeatuRSEB and VBFD (see Figure 4. 14 and Figure 4. 16) presented variations and variation points instead of the tradition or and Xor relationships present in others.
- FODA and FORM represents less readable feature model as their notations requires additional explanations. However, in terms of readability and simplicity, GP tends to be more concise and has less constructs with clearer connections and relationship. FeatuRSEB notation tends to be the simplest and the most adequate assessed notation as it has UML constructs (Gomaa, 2005).

4.4.2.3. Traceability

The models ability to evolve and adapt to changes is an imperative aspect in modeling SPL, as they are always prone to alteration and changes. Thus reliability on the feature model in terms of manageability, data consistency and traceability is a requirement for an efficacious system. Traceability to track down features is indispensable, as the feature model needs to evolve by means of adding, updating, extending or integrating new features or requirements (Berg & Muthig, 2005; Metzger et al., 2007).

- FODA, FORM, as well as GP lack to support product line adaptability and growth. Moreover, the above-mentioned feature notations fail to promote evolution and integration of features and requirement.
- While, even though FeatuRSEB and VBFD are more manageable in terms of integrating features and requirements, in which VBFD provides enhanced dealing with binding times, as per annotating relationships between features, and FeatuRSEB provides binding time, which are reuse-time and use-time binding. Still they don't advocate complete traceability.

This infers that the feature modeling notations taken into consideration have shortcomings when considering traceability. The feature modeling approaches falls short in providing traceability, ability to evolve and adapt, and data consistency at the same time.

4.4.2.4. Scalability

For every feature model, it is crucial to be scalable and expandable to serve large scale SPL systems, as most organizations requires enormous SPL systems, and thus scalable feature models (Berg & Muthig, 2005). However current feature models don't seem to provide feasibility when it comes to large scale systems (Lee et al., 2002; Riebisch, 2003).

- It is evident that FODA has glitches in terms scalability (Batory et al., 2006), in which it only serves small scale systems and isn't feasible in handling large scale systems and complex ones. FODA and FORM present a lack of scalability and extendibility. The feature models investigated doesn't present feasibility in terms of adding extensions or managing large complex scales. They lack ease of management in such a way it is too complex to be handled on a scale other than a small scale on their own.
- In order to achieve scalability, GP, FeatuRSEB and VBFM approaches might be scalable under the condition of being supported and analyzed automatically. CASE is a support tool that underpins feature modeling notations in terms of scalability and capability to modeling all features and any extensions.

It is important here to stress out the perceptible lack of independency of current FM notations when it comes to scalability and dealing with complex systems.

4.4.2.5. Articulacy

Ambiguities, uncertainties and misunderstandings create a blockage towards providing an articulated feature modeling notation that facilitates the perception of the SPL system proficiently.

- For an instance the lack of precision, as well as complexity and ambiguity resulting from the lack of interdependency between features exists in FODA and FORM (Van Gurp et al., 2001).
  The Or relationship doesn't exist, and alternative relationship isn't clear. Henceforth, Conflicts and ambiguities are present.
  Alternative feature are occasionally misrepresented, in which it creates conflicts and ambiguities by confounding it with optional or mandatory features (Metzger et al., 2007; Riebisch, 2003).
- Moreover, FeatuRSEB presents vagueness and confusion in relationships between features in terms of optional and the Or relationship, as well as Or and Xor relationship.
- VBFM and FeatuRSEB are also accompanies with ambiguities related to variants. In addition, there might present concomitance of the Or and Xor relationships with the mandatory and optional features in the generative programing model.

Having that said, the assessed feature models lacks to have an articulated and clear-cut semantics and notations. Thus the approaches explored don't lack inconsistent interpretations and uncertainties.

4.5. Automated Analysis of Feature Models

The realm of software product lines currently has an intensifying prominence, as per its proficiency to enhance software reuse and make the whole procedure more feasible. Ever since the very first establishment of feature models (Kang et al., 1990 Novak, & Peterson, 1990), the manual management, operation and handling of the feature models have always been challenging and error-prone mission.

The glitches in feature modeling approaches has been identified (refer to section 4.4), in addition to the need to support operations related to void features, invalid or partial configurations, lack of flexibility and simplicity in product line, optimization, and undetected anomalies (Batory, 2005; Benavides et al., 2010 2010; Cuevas, 2007; Mannion, 2002; Perez-Morago, Heradio, Fernandez-Amoros, Bean, & Cerrada, 2015 Bean, & Cerrada, 2015).

Consequently, as per knowing the importance of software product line systems, SPL specialists have proposed a vast amount of validation techniques to endorse its process (Amine, Mohamed, & Bellatreche, 2013 2013). Since these problems in feature models cannot be detected manually and on the sole dependence on the FM approach used, which is highly complex and difficult, analysis operations is needed. Thus, The automated analysis operation on feature models allows the direct validation and verification of the feature models in the SPL system.

Within here levitated the role of the automated support, which is responsible for the analysis of feature models in SPL. Consequently, specialists in SPL proposed various practices to deliver automated analysis of feature models. These proliferated automated analysis of feature models aims to deliver techniques, algorithms and tools that work towards an automated extraction of information from feature models, in which features, dependencies and any incompatibilities are recognized, analyzed and disentangled.

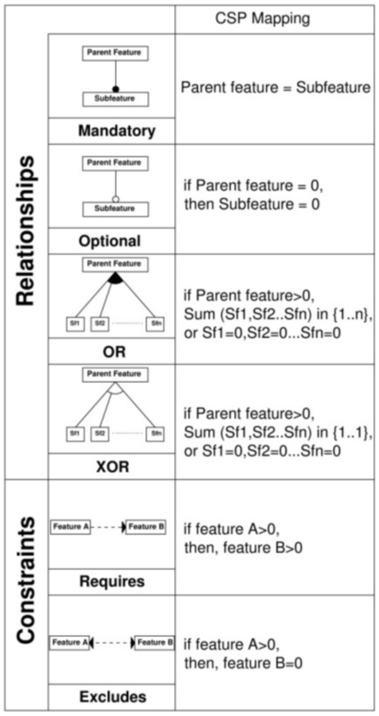
This section will provide a concise review of the foremost promising solvers and approaches in the perspective of the automated analysis of feature models.

Those proposed automated analysis approaches ranges from propositional logic and first order logic approaches (SAT solvers, BDD, Alloy…) (Mannion, 2002; Mendonça, 2009; Sun et al., 2005 & Wang, 2005), constraints programming (CSP solvers and its derivatives) (Batory et al., 2006 2006; White et al., 2009 & Benavides, 2009), description logic approaches, conceptual logic approaches, and heuristic solutions (Ad-hoc) (Batory et al., 2006; Kiniry, 2007) . In particular, this section will consider the solvers and approaches listed and explored as follows.

4.5.1. Constraint Satisfaction Problem solver

The constraint programing is made out of a set of techniques dealing with CSPs. Those techniques can be algorithmic and heuristic (Tsang, 2014).

Feature models are transformed into Constraint satisfaction problem "CSP" which is made of set of variables, coupled with their set of domains which are expressed by integers and interval values, and constrained by the set of constraints (see Figure 4. 17).



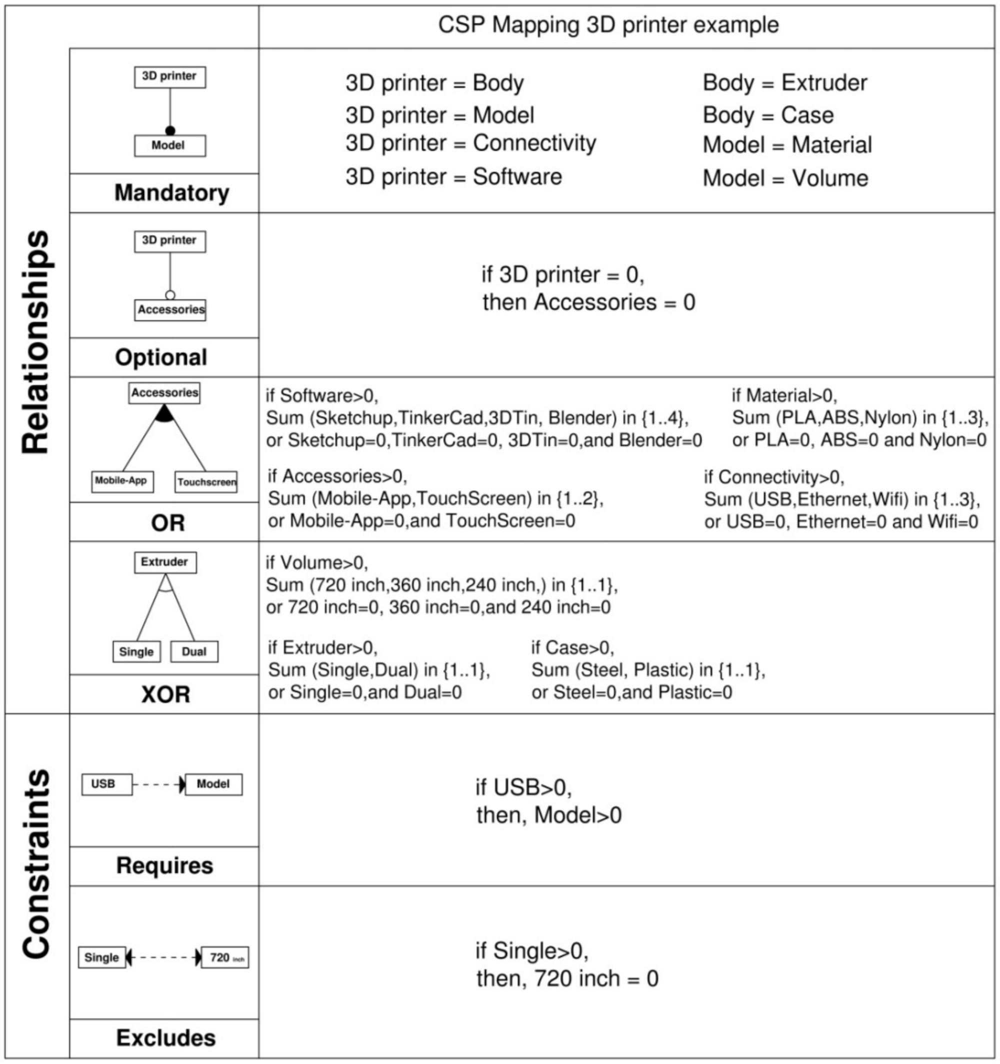
		CSP Mapping
Relationships	Mandatory	Parent feature = Subfeature
	Optional	if Parent feature = 0, then Subfeature = 0
	OR	if Parent feature>0, Sum (Sf1,Sf2..Sfn) in {1..n}, or Sf1=0,Sf2=0...Sfn=0
	XOR	if Parent feature>0, Sum (Sf1,Sf2..Sfn) in {1..1}, or Sf1=0,Sf2=0...Sfn=0
Constraints	Requires	if feature A>0, then, feature B>0
	Excludes	if feature A>0, then, feature B=0

Figure 4. 17 CSP mapping of feature model

A constraint satisfaction problem solver is a tool or software that formulates CSPs. These solvers works by considering and modeling the problems. The modeling elements are expressed as variables, domains and their constraints. The CSP solver aims to study the problem and search for the existence of any possible solutions following the next steps (Benavides, Segura, et al., 2006b):

- Features are mapped to CSP variables alongside the feature's domains according to the kind of support provided by the CSP solver, which is either dependent on TRUE or FALSE or 0…1.
- Dependencies and constraints are then considered. In respect to that, corresponding variables present.
- After identifying the variables, their domains and constraints, all the mentioned entities will be assigned a value in accordance to their dependencies for example parent feature = TRUE or Parent Feature = 1, which is governed by on the domain variables.

Figure 4. 18 clarifies the CSP mapping in accordance to the 3D printer exemplar designed and used throughout the whole thesis.


3D printer
Model
3D printer
Accessories
Accessories
Mobile-App
Touchscreen
Extruder
Single
Dual
USB
Model
Single
720 inch

		CSP Mapping 3D printer example

Relationships	Mandatory	3D printer = Body
		3D printer = Model
		3D printer = Connectivity
		3D printer = Software
		Body = Extruder
		Body = Case
		Model = Material
		Model = Volume

	Optional	if 3D printer = 0, then Accessories = 0
	OR	if Software>0, Sum (Sketchup,TinkerCad,3DTin, Blender) in {1..4}, or Sketchup=0,TinkerCad=0, 3DTin=0,and Blender=0 if Material>0, Sum (PLA,ABS,Nylon) in {1..3}, or PLA=0, ABS=0 and Nylon=0 if Accessories>0, Sum (Mobile-App,TouchScreen) in {1..2}, or Mobile-App=0,and TouchScreen=0 if Connectivity>0, Sum (USB,Ethernet,Wifi) in {1..3}, or USB=0, Ethernet=0 and Wifi=0
	XOR	if Volume>0, Sum (720 inch,360 inch,240 inch,) in {1..1}, or 720 inch=0, 360 inch=0,and 240 inch=0 if Extruder>0, Sum (Single,Dual) in {1..1}, or Single=0,and Dual=0 if Case>0, Sum (Steel, Plastic) in {1..1}, or Steel=0,and Plastic=0
Constraints	Requires	if USB>0, then, Model>0
	Excludes	if Single>0, then, 720 inch = 0

Figure 4. 18 CSP mapping of 3D printer feature model

4.5.2. SAT solvers

SAT solvers are propositional logic based analyses solvers. Propositional logic analysis PL, to begin with, uses propositional formulas that are made up of sets of symbols deciding the connections and constraints of the analyzed feature models. Figure 4. 19 demonstrates the propositional logic mapping of a feature model, which provides an illustration for the symbol used to decide the variables, domains and constraints (Mannion, 2002; Mendonca, Wąsowski, et al., 2009).

SAT solver is a tool which translate the feature model by means of propositional formula, to determine whether its Boolean expression is satisfiable or not, and reach a solution to the SAT.

In the majority of SAT solvers, the propositional formulas are entered as Conjunctive Normal Form CNF (Cook, 1971).

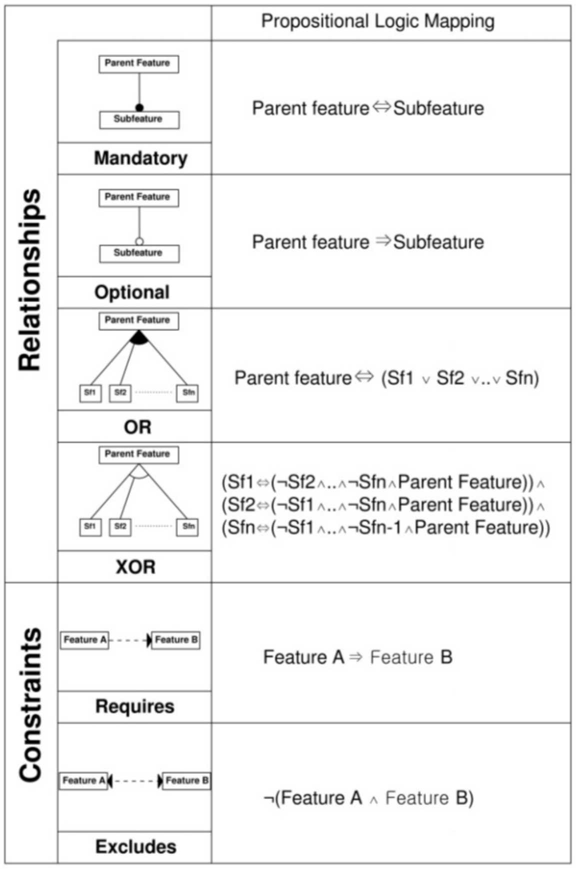

Propositional Logic Mapping
Relationships
Parent Feature
Subfeature
Mandatory
Parent feature ⇔ Subfeature
Optional
Parent feature ⇒ Subfeature
Sf1
Sf2
Sfn
OR
Parent feature ⇔ (Sf1 ∨ Sf2 ∨..∨ Sfn)
XOR
(Sf1⇔(¬Sf2∧..∧¬Sfn∧Parent Feature))∧
(Sf2⇔(¬Sf1∧..∧¬Sfn∧Parent Feature))∧
(Sfn⇔(¬Sf1∧..∧¬Sfn-1∧Parent Feature))
Constraints
Feature A
Feature B
Requires
Feature A ⇒ Feature B
Excludes
¬(Feature A ∧ Feature B)

Figure 4. 19 PL mapping of feature model

4.5.3. Binary Decision Diagram solvers

Binary Decision Diagram BDD solvers, similar to SAT solvers, are propositional logic based analysis solvers that translates propositional formulas into graphical presentations. Those decision diagrams represent the Boolean functions in the form of DAG, which consequently facilitates the automated analysis process of the feature model. BDDs are computational tools that functions by figuring out the satisfiability and deciding the algorithms that supports establishing feasible solutions.

Overall, when mapping feature models by means of propositional logic based analysis, this mapping varies depending on the solver used for the automated analysis. However the following approach is mostly considered (Benavides et al., 2009):

- Features are mapped to propositional formula variables alongside the feature's relationships which are mapped to smaller formulas in accordance to the feature model being analyzed.
- After identifying the variables, and any possible auxiliary variable resulting from mapped relationships, a final formula is conveyed which is the result of the conjunction of all the mentioned formulas and the constraints present in the feature model.
- The final stage lies on assigning the truth value to the variable of the root feature.

4.6. Concluding Remarks

During the course of this chapter, Feature modeling in SPL has been explored, to satisfy our intention in determining the glitches and shortcomings accompanying previous notations and approaches.

The literature review first of all presented an insight of definitions and terminologies of feature modeling, their different notations and constructs, to acquaint the reader with our exploration by means of systematic literature review.

Throughout the whole chapter, an exemplar of 3D printer product line is used as representation for feature models to express their various notations, constructs and specifications. Moreover, using the same exemplar throughout the whole chapter aided comparing and contrasting between various notations that were put under the lens of assessment.

Glitches and shortcomings of the current feature modeling notations are identified, wherein an assessment of selected FM notation is undertaken, which are FODA, FORM, FeatuRSEB, GP, and VBFD Notations. The Notations are critically assessed in terms of scalability, traceability, articulacy, comprehensiveness and visual presentation suitability. The findings are recapped as follows:

- Current FM notations are frequented with complexity and obscurity in relations and dependencies semantics. In this manner, there's a serious need to take into account further dependencies semantics to avoid misperceptions.
- Although there's a discrepancy in the type of visualization in FM notation and consequently a variation in the visual presentation, there are challenges in attaining an all-inclusive readable and simple construct as marked in FODA and FORM notations.
- The evaluated FM notations tend to be infeasible in terms of providing traceability, ability to evolve and adapt, and data consistency concurrently.
- Evidently speaking, current FM notations can't perform adroitly in large scale and complex systems and have discernible scalability problems.
- The assessed feature models are deficient in terms of providing articulated notations. Furthermore they exhibit some failures in terms of capturing some existing semantics among the model parameters.

Subsequently, this leads us to our next part of the dissertation targeting to explore the realm of probabilistic modeling which attempts to overcome the problems discussed in this chapter.

"It should perhaps be noted that the choice of variables in terms of which a given problem is formulated, while a seemingly innocuous step, is often the most crucial step in the solution."

-Callen, 1985

Part IV

Modeling

Chapter 5

Modeling under Uncertainty

In information theory, information integrated among a set of variables are a measure of the variables' capability of reducing the systems' uncertainty (Kullback, 1968). Variables with higher uncertainty contain more information than variables with lower uncertainty. This notion motivated scientists to quantify information as per its probability weight. Claude E. Shannon (1949) noticed that $I(p) = -\log_b(p)$ such that p is the probability weight of a certain variable and I is the measure of information.
This notion reveals three main properties of information:

1. Information is always none negative quantity,
2. Variables with Certain probability weight don't provide any new information,
3. Information measure of Independent Variables is additive.

Clearly, to capture the semantic of any variable; it's important to anticipate its uncertainty level. The third property of information is effective, when we learn about belief model, in which model variables are mutually independent.
To practically understand the dependency behavior and relationship among a set of variables, its encouraged to structure data model defining the correlation among variables, and quantifying the degree of belief for each variable. Moreover, data models are very effective to predict the outcome of variables aggregation, and trace

the dependency flow among the model parameters.

In SPLE, relationships among variables are captured via logical notation known as Feature Model. Feature models are information models used to identify products of product line. The information embedded in the model, are denoted as variables within different dependency contexts and crosstree constraints among it. Variables can either be core variables (which will be referred by core features throughout this chapter) with definite probability weight to ensure its existence in all products configurations, or varied variables (which will be referred by variables throughout this chapter) with different uncertainty level. Observing core feature with probability weight equal one doesn't give us any new information about the expected products' functionality.
On the contrary, observing a variable would increase the model uncertainty; as a result provide new information about the expected product identity. This conceptual notion is mathematically valid and semantically mapped with the first and second property of the information as been identified in the information theory.
Due to its uncertainty nature, variables enrich FM with information. The more variables the model has, the more information we can obtain from this model. Information can be seen as non-functional or functional qualities of the model product line. This implies that, more variables potentially bring about more products. When modeling in SPLE, the uncertainty measure of any variables is due to its interaction with the model parameters. The degree of uncertainty of any variable is a representation of our intuitive belief about this variable probability when we start constructing the model.

To capture the relationships and uncertainty measure of the model parameters; We introduce Bayesian Belief Network BBN as framework to construct a variability model, that doesn't only capture the logical dependency among features, but also anticipate the uncertainty measure of involved features. Providing belief model help us to understand the information provided in the model and aids in the reasoning process.

This Part of the dissertation will profoundly explore the semantic of FM in SPL, traced down by an overview to our proposed Bayesian belief feature model. Moreover, this part will draw out the semantic equivalence between the two distinct models as well as define the mapping rules amongst both.

5.1 The Notion of Feature Model Semantic

Feature Model FM is information model (Benavides et al., 2010) used to identify products in product line (Batory, 2005). Feature Model (FM) consists of features and relationships among them. In Feature model, features are typically a distinctive visible attribute used to indicate quality in the product functionality (Kang et al., 1990). In Addition, relationship in Feature Model can typically be defined as a class of dependency that indicates the interaction between at least two different features, see Figure 3. 4 for an example of feature model. FM is known as variability model, such that variables appear to interact throughout different dependency channels, allowing the model to obtain different functionality each time we have a valid and complete explicit set of variables.

Current practices compose features and dependency relationships into a hierarchical graphical representation called Feature Diagram. In Feature Diagram FD, composition rules that specify mutual dependencies and mutual exclusion between features (Kang et al., 1990) appear as crosstree constraints (Benavides et al., 2010).

The main types of relationships to group a set of features can be identically categorized as; Or, Optional, Mandatory, and Alternatives; which are used to maintain the dependencies level between a compound feature and sub feature (Batory, 2005).

5.1.1. Feature Model Prominence

The key importance of FM is to elucidate the requirement space, in order to support the development process and the reuse of notation. The two main components of any feature model are Feature Diagram and Composition Rules (Benavides et al., 2010).

In the development process, FM could be used as a knowledge domain to indicate what has to be parameterized in other models (Kang et al., 1990).

Feature Diagram is relatively easy to read rather the Model, which is solely composed of composition rules (Kang et al., 1990). Therefore, Feature Diagrams are widely adopted as accepted representation denoting all members of a given Product line (Benavides et al., 2010).

To construct a Feature diagram, a data analysis should be first conducted to quantify features and its corresponding functionality, with extensive emphasize on proposed requirements of the anticipated products. The quantification process determines any identified dependencies among different features. Moreover while identifying features dependencies; grouping contexts could be introduced as a resolution of multiple same level dependency channels among prior and posterior features. The obtained information defines the problem domain knowledge.

Once the knowledge domain is clearly identified by naming all features and the dependencies among these features, a feature diagram can be constructed in which the semantic of knowledge domain is fully captured.

5.1.2. Relationships in Feature Model

A typical relationship between features in FM is a direct dependency between two features or more. A *consist-of* relationship demonstrates a logical grouping of dependent features (Kang et al., 1990).

Theoretically, any kind of relationship could be defined to capture the dependency semantic between a set of features (P. P.-S. Chen, 1976).

In the interest of Software Product line Engineering SPLE, four structural relationships are used to capture the dependencies between features that are:

- Mandatory AND Dependency.
- Inclusive OR Dependency.

- Exclusive OR Dependency.
- Optional Dependency.

And two composition rules typically named Crosstree Constraints that are:

- Mutual include.
- Mutual Exclude.

In the following we are going to carefully identify the semantic of each relationship.

- **Mandatory AND dependency:**

  Mandatory AND dependency is a direct dependency between a compound feature and subfeature, in which subfeature must be included in the system specification wherever its compound feature is part of the system specification.

  Let $\{f_1, f_2\}$ be of a set subfeatures in mandatory AND dependency with compound feature f_c.
  The truth semantic of this dependency context holds only when $f_c := f_1 \cdot f_2$ such that *f* can have a binary value of zero or one.

- **Inclusive OR Dependency:**

  A set of subfeatures is composed by an Inclusive Or Dependency with a compound feature, enforces that at least one subfeature must be included in the system specification wherever the compound feature is included.

  Let $\{f_1, f_2\}$ be of a set subfeatures in Inclusive OR dependency with compound feature f_c.
  The truth semantic of this dependency context holds only when $f_c := f_1 + f_2$ such that *f* can have a binary value of zero or one.

- **Exclusive OR Dependency:**

  A set of subfeature are composed together with a compound feature in cardinality-based dependency, illustrating that only one subfeature must be included in the system specification, whenever the compound feature is included.

  Let $\{f_1, f_2\}$ be of a set subfeatures in Exclusive OR dependency with compound feature f_c.
  The truth semantic of this dependency context holds only when $f_c := \overline{f_1} . f_2 + f_1 . \overline{f_2}$ such that *f* can have a binary value of zero or one.

- **Optional dependency:**

  Optional dependency is a direct dependency between a compound features and subfeature, in which, a subfeature can but doesn't need to be part of the system specification whenever the compound feature is included.

  Let $\{f_1, f_2\}$ be of a set subfeatures in Optional dependency with compound feature f_c.
  The truth semantic of this dependency context holds only when $f_c := (\overline{f_1} + f_1) + (f_2 + \overline{f_2})$ such that *f* can have a binary value of zero or one.

- **Mutual Include:**

  Mutual Include indicates all optional and alternative features that must be included in the system specification whenever a given feature is included.

  Let f_1 and f_2 be two features such that, f_1 mutually include f_2 when $f_1 \rightarrow f_2 := \overline{f_1} . \overline{f_2} + f_2$, in a manner *f* can have a binary value of zero or one.

- **Mutual Exclude:**

  Mutual Exclude indicates all optional and alternative features that must be excluded from the system specification whenever a given feature is included.

  Let f_1 and f_2 be two features such that, both features are in mutual exclude context whereas $f_1 \leftrightarrow f_2 := \neg(f_1 . f_2)$, such that *f* can have a binary value of zero or one.

5.2. The Need for New Model

The key innovation of Feature Model is to create a representation that specifically captures the functionality of products, throughout exploiting the semantic existing among the product components. Current methodologies, institute a good framework in which basic representation is provided to capture the meaning of model components, and including direct dependencies among features; while identifying possible boundaries in the product line. Nevertheless, these representations mostly exhibit shortcoming in the interest of non-direct interaction and the dependency semantic among model components. Also, the predefined composition rules are usually orienting around the direct dependencies between features in Feature Model.
Knowing that; the complete feature attribute is naturally inherited by its ancestors and bounded by its mutual dependencies as well as its descendants, the need of understanding the dependencies flow in the system specification has emerged as a key issue (Apel, Batory, Kästner, & Saake, 2013a). According to (Benavides et al., 2010; Kang et al., 1990), current representations exhibit obvious limitation in this regard, and tend to leave some semantic to common sense, which usually become less obvious and get lost overall complexity.
Moreover, current feature models usually assign equalized weight assignments for all model components without indicating the likelihood of each component, or allowing better understanding of the dependency flow; to consider any latent implication raised by indirect dependencies.

Proposing different weights values for feature model parameters, is a promising technique aiding to quantify the intended implication of involved components.
By identifying the implication weight of each component in the feature model, we advance our belief about the components interaction and the truth behavior of the feature model.
By tracing the dependencies flow, and anticipate the truth assumption of each parameter, we would be able to enhance the feature model performance throughout a possible rearrangement of features integration, or by reducing some parameters as per its probability weight, alongside with its integration and implication on the model belief. Any reduction in feature model would reduce the reasoning problem space, leading to an efficient reasoning process.
In addition, while advancing our belief about the model behavior, we will be able to efficiently advance our design even in latter stages, when satisfying features requirements.
For instance, if we observe a feature with a relatively low probability weight, we can conclude that the chances of including this feature in a valid configuration would be minimal. After that we might need to revise the feature attribute, such as; cost, functionality, added complexity, size. By recalling the initiated knowledge domain, and product line preference; we might decide either to omit this feature from the model, consequently we reduce cost and problem size in addition to improve the reasoning process. On the contrary, we might decide to increase the probability weights of this feature (due to preferred functionality, variability) by either incorporate some composition rules calling this feature or a potential rearrangement of features integrations.
In both scenarios the belief model helps us to improve our design and utilize the usage of model parameters.

The need of incorporating new components to the core components, or even remove some variables without devastating the overall semantic, is another reason to start thinking of developing a more dynamic modeling framework that gives the designer a higher degree of freedom during the modeling stage.

Moreover, we take into consideration that tackling a problem from the problem space into a designed model is also an art as much as a science. Therefore new modeling techniques should be always considered to pair different solution dialects for different problems contexts.

To conclude, the demand for a new model is pinpointed for the following motives:

- The need for capturing the actual implications of the existing features, and quantifying of the occurrence likelihood of each feature.
- The need to quantify non-direct entanglements among model parameters, throughout a determination of the truth flow and uncertainty variation among different contexts and parameters.
- The necessity to support the automation analysis and reasoning process, by exploiting the dependency strength and inferring the actual implication of the embedded crosstree constraints.
- The aspiration of enhancing the model graphical representation, by the use the variation of color to project variation in the uncertainty measure.
- The need to inform decisions when trying to derive new products in the product line.
- The desire of having dynamic model that enable the user to easily incorporate new features or remove some existing features from the original design, and yet still can quantify the added implication of the new change.

5.3 Bayesian Modeling

Bayesian Belief Network "BNN" is famously known for its conveyance in artificial intelligence as an attempt to model. BBN is thus a modeling approach which is basically a directed acyclic graph that merges between both probability and graph theory. Thus, BBN is an all in one approach to model, as well as, reason and handle uncertainties at the same as been equipped with modeling and probabilistic methods.

In addition to that, BBN provides an advancement in the realm of information since this belief network supports both qualitative and quantitative information captured by means of a rationale technique. Throughout the usage of BNN, any new information can swiftly be incorporated and added up to the information base as soon as it's present. However when it comes to dealing with conditional uncertainties and the probabilistic aspect of the BBN, determination and reasoning is required. That being said, the BBN is consequently associated with an extent of changeability and variability when considering any model being developed by means of BBN approach.

Due to its precedence in terms of dealing with unpredictability, variabilities and uncertainties, BBNs seems to be the most favorable and adequate approach to run through the outcome of the highly tendency of occurrence in future scenarios.
The BBN functions by means of probabilistic inference, in which it computed the probabilistic weighting of the considered variables in accordance to information about other related variables and their conditions and contexts.

As been stressed out earlier, the prominence of BBN lays within the capability of handling information coming from subjective judgment as well as objective ones.
So, in terms of objective data scarcity, it's possible to use subjective data to form out the initial material to be assessed and weighted probabilistically and later on, any additional beneficial information could be encapsulated in the BBN and thus provide an update and rebuild up the probabilistic outcome.

In conventional basic FM, a set of observed components always exist as a core features, as well as, a set of varied components that usually alters from specification to another within a context of dependencies. The more variables we have the higher uncertainty level we reach. This is usually a desired aspect in any FM, but on the downside this will raise range of complications regarding scalability, satisfiability, visibility.

In feature modeling, the existence likelihood of any child subfeature is conditioned by the existence state of its parent compound feature. The truth flow is directed from parent compound features to child subfeatures in a casual flow. Thus, can be captured using Bayes theorem, in which the truth probability of any posterior event is conditioned by the truth probability of the prior event.

Both BBN and FM notations are directed acyclic graph in which the occurrence of any posterior event is restricted by the occurrence of its prior event. BBN modeling is effective when inferring a consistent conclusion from the given contexts, in which it helps to inform decisions accordingly.

In addition, integrating new features whenever available is a core advantage of BBN, while also subjective assignment of the truth measure among the model parameters is possible throughout different design stages of the model.

Moreover, BBN is an assimilation between graph theory and probability theory that has been successfully implemented and evolved in science and engineering to provide, not only a modeling framework, but also a predictive approach in which the causality of events can be quantified and predicted easily.

We introduce Bayesian Belief Network based modeling technique to model basic FM as a pioneering comprehensive framework to allow users to reason and model about uncertainty.

Our approach is aided to:

1. Improve the design performance.
2. Capture the existence semantic between features throughout a comprehensive belief network.
3. Reduce number of parameters in a given problem space.
4. Present Scalable prediction of features latent implication and occurrence likelihood.
5. Quantify the crosstree constraints implication.
6. Reason and satisfy the resultant model with a sense of uncertainty.
7. Exploit component usage and influence.

Basic FM indicates what needs to be configured into the model and how the semantic of configuration should be mapped (Kang et al., 1990). Therefore, we use existing FM as our knowledge domain due to its simplicity and wide adoption.

In the next we introduce our BBN based modeling framework and define the mathematical semantic of each component to assure that at least all semantic provided by Basic FM are true and hold in the new proposed model.

5.3.1. Bayesian Belief Feature Model BBFM

Bayesian belief feature model BBFM is a belief representation of the knowledge domain, such that a probability weight is assigned for each feature, to quantify its implication on other model parameters. These assignments are used to specify the probability of satisfying the existing semantic of the involved dependency contexts. The identified probability weight helps to optimize the model behavior while reasoning, also anticipates the probability weight of any given feature.

BBFM is composed of a set of dependency contexts that captures the existence semantic among set of variables.

The developed model is inspired by the work of (Shwe et al., 1991) who pioneered a Noisy-OR gate to model Quick Medical Reference QMR diagnostic decision support tool. His model has gained a huge success by reducing number of involved parameters significantly.

The developed Bayesian belief model BBM is composed of four main dependency contexts namely; Bayesian Conjunction, Bayesian Disjunction, Bayesian Exclusive Disjunction, Bayesian Tautology, in addition to direct mutual include and mutual exclude dependency; fusing features constraints.

We can structure Bayesian Belief Feature Model BBFM by grouping each set of variable within its dependency context in cardinal format, allowing the truth-value to flow within dependencies context evidentially. The structure of the dependency context must be valid and corresponding with the semantic of the equivalent Basic feature model FM.

Figure 5. 9 illustrate a translated model from FM (refer to Section 3.3 and Figure 3. 4) into BBFM of our designated example of 3D printer.

Definition 5.0. We define Bayesian Belief Feature Model BBFM, as a set of interconnected dependency contexts, such that each dependency context must be categorized as one the following Bayesian dependency context; Bayesian conjunction, Bayesian disjunction, Bayesian exclusive disjunction or Bayesian tautology, And two mutual dependency; Require and Exclude.

Different probability weights are distributed among belief model, providing a measure of the parameters' expected affect.

This notion is very crucial to give the designer the ability to emphasize on some attributes rather than others, to meet some non-functional properties, or to reduce the influence of some constraints..etc.
It also helps in latter development process, in which the need of incorporating a new feature, or even drive the dependency flow between features, is emerged; to obtain a desirable efficiency in the reasoning process.

In the following subsections, the mathematical semantic of the developed Bayesian model will be explained thoroughly.

Definition 5.1. Bayesian dependency context; is a dependency configuration composed of set of features f and dependency function θ. Such that θ defines the dependency semantic existing among compound feature f_c and all its adjacent subfeatures$\theta:\{f_c, f_{s1} \dots f_{sn}\}$.
Each feature f has probabilistic truth assumption C_f denoting the entanglement with other features via crosstree constraints, in the Bayesian Belief Feature Model.

Where θ has probabilistic truth assumption γ denoting the probability of satisfying the induced semantic in accordance with the model belief.

5.3.1.1. *Bayesian Conjunction*

Definition 5.2. A set of subfeatures $\{f_s, \ldots, f_{sn}\}$ is in Bayesian Conjunction $\theta_\bullet$ with compound feature f_c, such that $\theta_\bullet : \{f_c, f_{s1} \ldots f_{sn}\}$.
In which, the truth assignment of the compound feature is determined by the truth assignment of all subfeatures, demanding all subfeatures must be true in all system specifications wherever the compound feature is true.

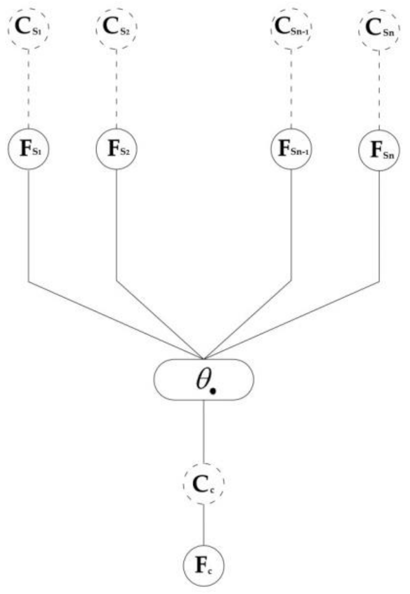


Figure 5. 1 Graphical Representation of Bayesian Conjunction

Lemma 5.1. Let $\theta_\bullet : \{f_c, f_{s1} \ldots f_{sn}\}$ be generated via the Definition 5.2. Where $\{f_{s1}, \ldots, f_{sn}\}$ is a subset of the set of all subfeatures. Such that,

$$\{f_{s1},..,f_{sn}\} \in \{all\ \ subfeatures\}.$$

Assuming that $\{f_{s1},..,f_{sn}\}$ are observed subfeatures with valid truth assumption, such that,

$$P(f_s \mid c_s) = \omega_s.$$

We define the probabilistic truth assignment of obtaining a valid $\theta_{\bullet}$

$$P(\theta_{\bullet} \mid f_{s1},..,f_{sn}) = \begin{cases} 0, & \prod_{s=1}^{n} P(f_s \mid c_s) = 0 \\ \gamma, & \prod_{s=1}^{n} P(f_s \mid c_s) > 0 \end{cases}$$

Proof and Semantic Proof is by semantic check; Definition 5.2 demands that all subfeatures must be true to evaluate a truth assignment for the conjunction function.

Suppose we have set of observed subfeatures Lemma 5.1 evaluates the truth assumption of observed subfeatures, such that if any observed subfeature f_s has a probabilistic truth assignment $P(f_s \mid c_s)$ equal zero.

This will directly result the product $\prod_{s=1}^{n} P(f_s \mid c_s)$ to be zero, which in its turn, disqualify the conjunction function of being true and set its probability to zero.

Otherwise, if no observed subfeature had violated the conjunction requirement, *i.e* all observed subfeatures had truth assumption greater than zero, $\theta_{\bullet}$ is yet valid and have a probabilistic truth assignment equal γ ;

$$P(\theta_{\bullet} \mid f_{s1},..,f_{sn}) = \gamma$$

Corollary 5.1. Let f_c be a compound feature that's generated via Definition 5.1 with a valid dependency function. We define the probabilistic truth assignment of f_c

$$P\left(\overline{f_c} \mid c_c, \theta\right) = \left(1-\omega_c\right)\left(1-\gamma\right)$$

Similarly,

$$P\left(f_c \mid c_c, \theta\right) = 1 - P\left(\overline{f_c} \mid c_c, \theta\right)$$

Proof and Semantic Corollary 5.1 demonstrates that a compound feature f_c can only be true; if its truth assumption generalization and the truth assumption specialization [3] of the adjacent observed subfeatures f_s, exhibit truth-value.

Moreover, this notion implies that; the truth assignment for any feature in BBFM is actually determined with two major flows of dependencies; which are global dependencies and local dependencies:

- In global dependencies, we evaluate the casual dependency flow of observed features. Such that, if one of the observed features exhibit unsatisfiable crosstree constraint, this might jeopardies the truth assumption of its compound feature.
- In local dependencies, we evaluate the truth assumption of succeeded features and the dependency function, which can be seen as an evaluation of the feature generalization. Such that, if the compound features of feature *f* was assigned zero probabilistic weight, this will automatically disqualify feature *f* of having a valid truth assignment.

[3] feature *f* generalization is the evidential dependency's flow of feature *f*.
feature *f* specialization is the casual dependency's flow of feature *f*.

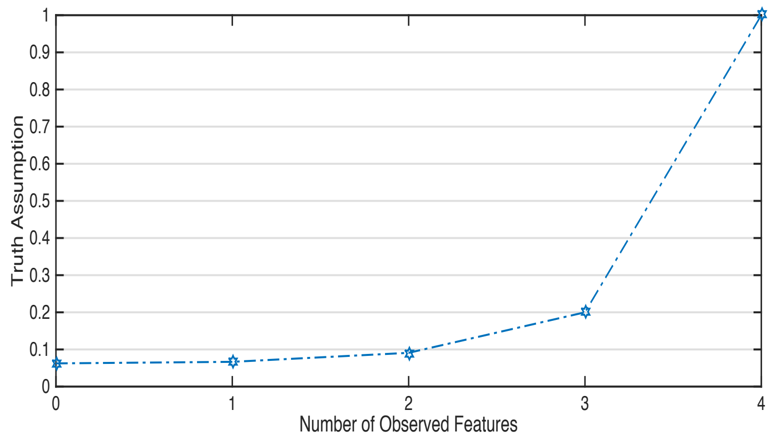


Figure 5. 2 Truth Assumption change in Bayesian Conjunction

Let $\theta_{\bullet}(f_c, f_1..f_4)$ be a Bayesian conjunction dependency context such that, four subfeatures are grouped in Bayesian conjunction dependency with a compound feature.

Figure 5. 2 above demonstrates the belief change of Bayesian conjunction functions' truth assumptions γ versus the number of observed subfeatures. Observing subfeature with a valid truth assumption would increase our belief about the truth assumption of the dependency function. γ Can only obtain a truth assignment equal one, only and if only all involved subfeatures were observed with a valid truth assumption.

5.3.1.2. Bayesian Disjunction

Definition 5.3. A set of subfeatures $\{f_s, \ldots, f_{sn}\}$ is in Bayesian Disjunction θ_+ with compound feature f_c, such that $\theta_+ : \{f_c, f_{s1} \ldots f_{sn}\}$.

In which, the truth assignment of the compound feature is determined by the truth assignment of at least one subfeature, allowing that, one subfeatures can satisfy the truth semantic of the Bayesian Disjunction θ_+.

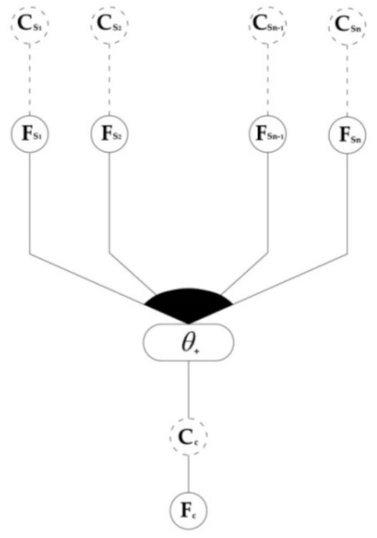


Figure 5. 3 Graphical Representation of Bayesian Disjunction

Lemma 5.2. Let $\theta_+ : \{f_c, f_{s1} \ldots f_{sn}\}$ be generated via the Definition 5.3. Whereas $\{f_{s1}, .., f_{sn}\}$ is a subset of the set of all subfeatures. Such that,

$$\{f_{s1}, .., f_{sn}\} \in \{all\ \ subfeatures\}$$

Assuming that $\{f_{s1}, .., f_{sn}\}$ observed subfeatures with known truth assumption, such that,

$$P(f_s \mid c_s) = \omega_s$$

We define the probabilistic truth assumption of obtaining valid θ_+

$$P(\theta_+ \mid f_{s1}, .., f_{sn}) = \begin{cases} \gamma, & \sum_{s=1}^{n} P(f_s \mid c_s) = 0 \\ 1, & \sum_{s=1}^{n} P(f_s \mid c_s) > 0 \end{cases}$$

Proof and Semantic Proof is by semantic check, Definition 5.3 requires that at least one subfeatures must be true to evaluate a truth assignment for the disjunction function.

Lemma 5.2 evaluates the truth assignment of the observed subfeatures, such that if any observed subfeature f_s has a probabilistic truth assignment $P\left(f_s \mid c_s\right)$ greater than zero this will directly increment $\sum_{S=1}^{n} P\left(f_s \mid c_s\right)$.

As a result, the sum will have a truth assumption greater than zero, which in its turn, qualifies the disjunction function to be true with probabilistic truth assumption equal 1.

Otherwise, none of the observed subfeatures succeed to exhibit truth assumption, the truth assumption of the disjunction function θ_+ is still due till we are able to determine the truth value of remaining subfeatures, hence γ .

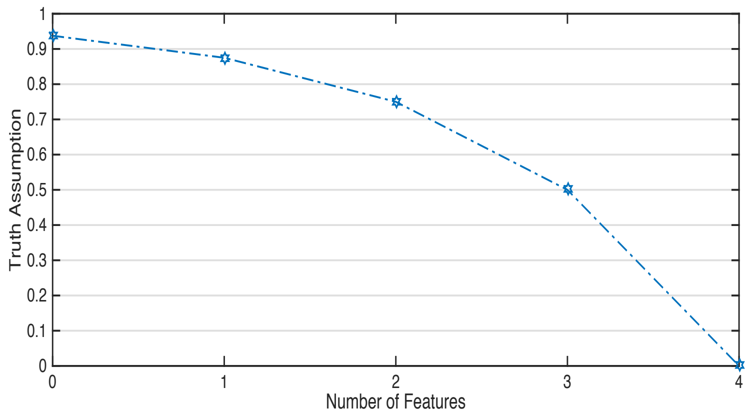


Figure 5. 4 Truth Assumption change in Bayesian Disjunction

Let $\theta_+\left(f_c, f_1 .. f_4\right)$ be a Bayesian Disjunction dependency context such that, four subfeatures are grouped in Bayesian disjunction dependency with a compound feature.

Figure 5. 4 above demonstrates the belief change of Bayesian disjunction functions' truth assumptions γ versus observing a number of features with invalid truth assumption. Observing subfeature with an invalid truth assumption $\omega = 0$, will decrease our belief about the truth assumption of the dependency function.

When observing that a given subfeature exhibit invalid truth assumption, this observation implies a reduction in the truth domain of γ and by default decrease the truth assumption value of the disjunction functions, the more invalid features we observe; the lower truth assumption we achieve until we observe that all involved subfeatures exhibit invalid truth assumption, in which the Bayesian disjunction function will be semantically invalid with truth assumption equal zero.

5.3.1.3. *Bayesian Exclusive Disjunction*

Definition 5.4. A set of subfeatures $\{f_s,...,f_{sn}\}$ is in Bayesian Exclusive Disjunction $\theta_{\oplus}$ with compound feature f_c , such that, $\theta_{\oplus}:\{f_c,f_{s1}...f_{sn}\}$.

If the truth assumption of the compound feature is determined by the truth assignment of one and only one subfeature, in which one subfeature must be true to satisfy the truth semantic of the Bayesian Disjunction $\theta_{\oplus}$, whereas all other subfeatures must be excluded from the system specification.

Graphically Bayesian Exclusive Disjunction is represented as follows;

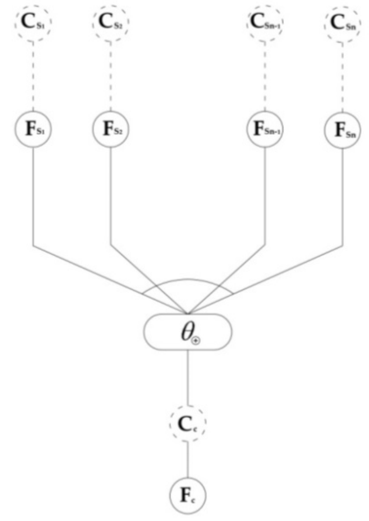


Figure 5. 5 Graphical Representation of Bayesian Exclusive Disjunction

Lemma 5.3. Let $\theta_{\oplus} : \{f_c, f_{s1} \ldots f_{sn}\}$ be generated via ***Definition 5.4.*** Whereas $\{f_{s1}, .., f_{sn}\}$ is a subset of set of all subfeatures. Such that,

$$\{f_{s1}, .., f_{sn}\} \in \{all \;\; subfeatures\}$$

Assuming that $\{f_{s1}, .., f_{sn}\}$ observed subfeatures with known truth assignment $P(f_s \mid c_s) = \omega_s$. We define the probabilistic truth assignment of obtaining valid $\theta_{\oplus}$

$$P(\theta_{\oplus} \mid f_{s1}, .., f_{sn}) = \begin{cases} \gamma, & \dfrac{\forall P(f_s \mid c_s)}{\sum_{s=1}^{n} P(f_s \mid c_s)} = 1 \\ 0, & otherwise \end{cases}$$

Proof and Semantic Proof is by semantic check, Definition 5.4 demands that one and only one subfeatures must be true to evaluate a truth assignment for the exclusive disjunction function.

In Lemma 5.3, we evaluate the truth assignment of the given subfeatures, such that if the probabilistic truth assignment of a subfeature f_s over the sum of the probabilistic truth assignments for all observed subfeatures $P(f_s \mid c_s)$ was exactly equal one. This means only one subfeature has a truth assignment greater than zero, concluding the observed subset doesn't violate the induced semantic of Definition 5.4.

Therefore the exclusive disjunction function is yet valid with probabilistic truth assignment equal γ.

Otherwise, if the given subset failed to meet the imposed semantic *i.e* more than one subfeature had a truth assignment greater than zero, thereafter, $\theta_{\oplus}$ is not true and has a probabilistic truth assumption equal zero.

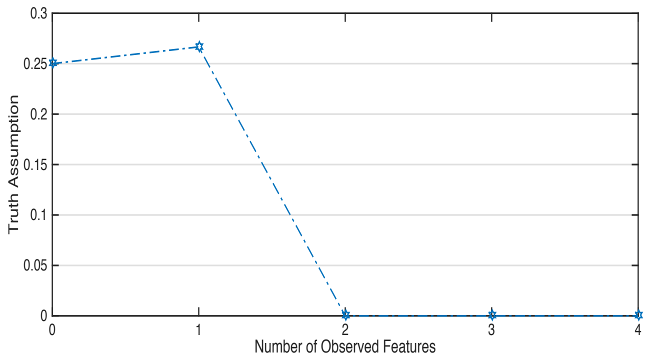


Figure 5. 6 Truth Assumption change in Bayesian Exclusive Disjunction

Let $\theta_{\oplus}\left(f_c, f_1..f_4\right)$ be a Bayesian exclusive disjunction dependency context such that, four subfeatures are grouped in exclusive disjunction dependency with a compound feature.

Figure 5. 6 above demonstrate the belief change of Bayesian exclusive disjunction functions' truth assumptions γ versus the number of observed subfeatures in a given subset. The highest truth assumption is achieved when only one subfeature observed and this subfeature exhibit a truth assumption.
If none of subfeature were observed we still obtain a truth assumption as we still have the possibility to obtain one valid subfeature. If more than one subfeature is observed with a truth assumption value greater than zero, this will violate the semantic of exclusive disjunction function, hence disqualify the dependency context of being valid and set γ to be zero.

5.3.1.4. Bayesian Tautology

Definition 5.5. A set of subfeatures $\left\{f_s, \ldots, f_{sn}\right\}$ is in Bayesian Tautology $\theta_{\top}$ with compound feature f_c ,such that $\theta_{\top}:\left\{f_c, f_{s1} \ldots f_{sn}\right\}$.
In which, the truth assignment for the set of subfeatures is actually doesn't affect the truth assumption for the tautology function; therefore the tautology function $\theta_{\top}$ is always satisfied.

Graphically Bayesian tautology is represented as follow;

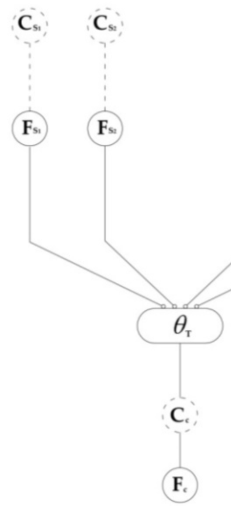


Figure 5. 7 Graphical Representation of Bayesian Tautology

Lemma 5.4. Let $\theta_T : \{f_c, f_{s1} ... f_{sn}\}$ be generated via the ***Definition 3.5*** ,whereas $\{f_{s1},..,f_{sn}\}$ is a set of all subfeatures. Such that,

$$\{f_{s1},..,f_{sn}\} = \{all\ \ subfeatures\}$$

Assuming that $\{f_{s1},..,f_{sn}\}$ are observed subfeatures with a truth assignment $P(f_s \mid c_s) = \omega_s$. The truth assumption of θ_T

$$P(\theta_T \mid f_{s1},..,f_{sn}) = 1$$

Proof and Semantic The proof is by semantic check, Definition 5.5 guarantees all problem space and any observation is in fact, a valid truth domain for the tautology function.

In Lemma 5.4, we evaluate a truth assumption for the tautology function for any set in the problem space.

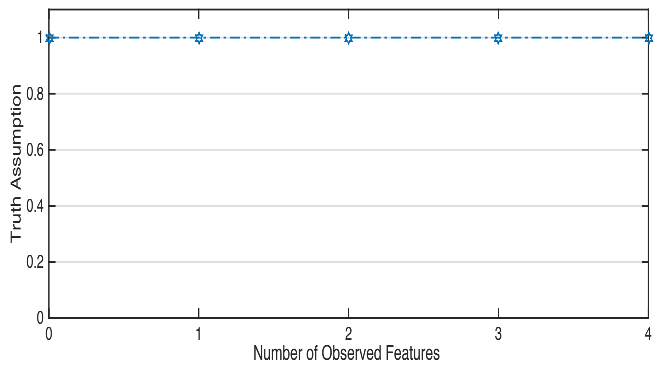


Figure 5. 8 Truth Assumption value in Bayesian Tautology

Let $\theta_{\mathsf{T}}\left(f_c, f_1..f_4\right)$ be a Bayesian tautology dependency context such that, four subfeatures are grouped in Bayesian tautology dependency with a compound feature.

Figure 5. 8 above demonstrates the belief of Bayesian tautology functions' truth assumptions is always equal one.

This implies that the tautology function is satisfied under any observation.

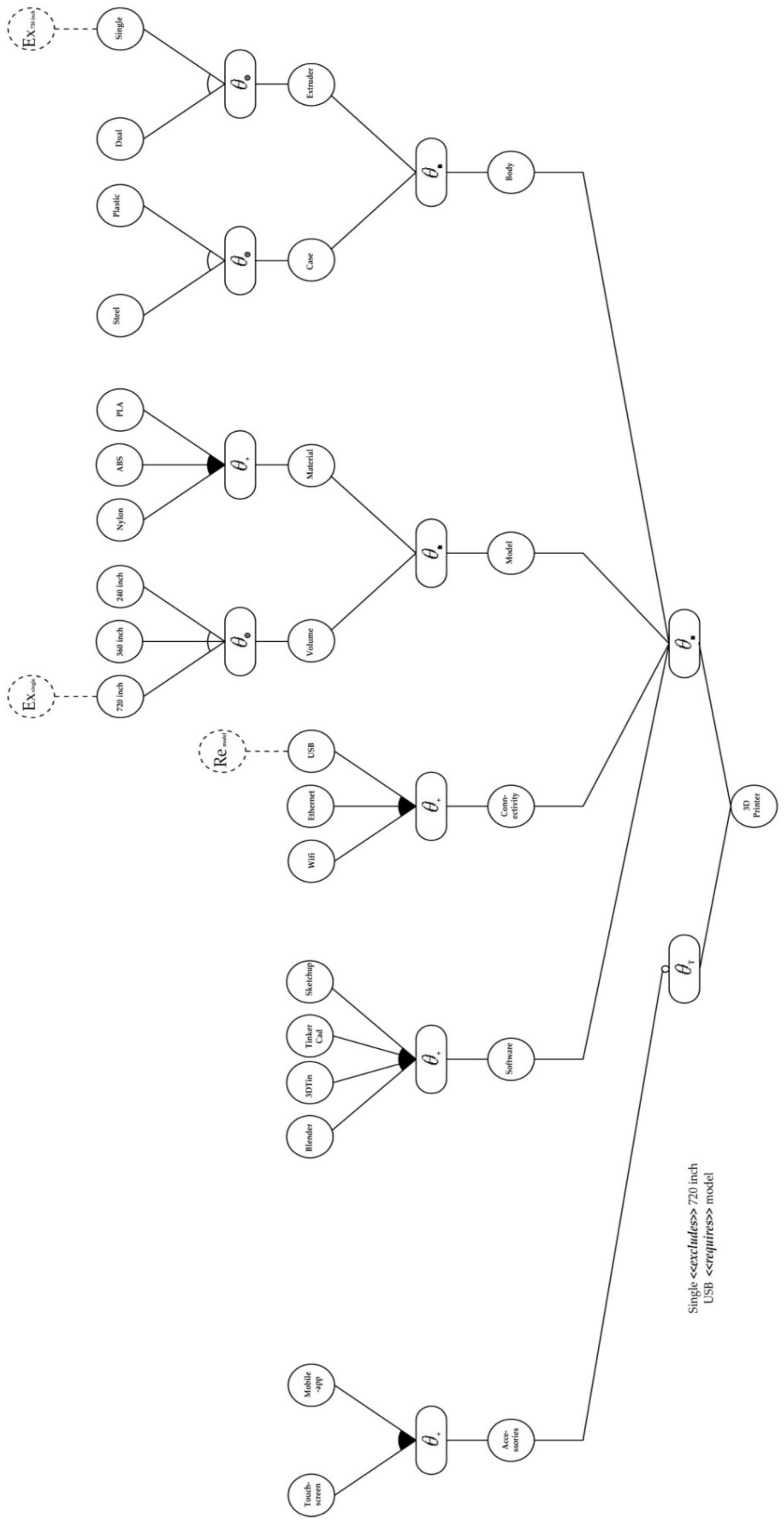


Figure 5. 9 Translation from 3D Printer FM into 3D Printer BBFM

5.4. Measuring the Uncertainty Level

Like any belief model, Bayesian Belief Feature Model BBFM quantifies the truth assignment for its parameters; throughout a collection of evidences and conclusions, which forms the belief model.

BBFM could potentially contain some core parameters namely core features, these must subsist in all valid configurations. A set of core features could derive the belief flow in BBFM. Knowing it has to be satisfied among system configurations; might imply a certain belief distribution all over the model.
This of course, would propose a truth assumption change amongst the model, by increasing or decreasing the truth assumption weight for other variables.

The aforesaid does not only give a better understanding of system parameters truth flow, but also it gives a possible truth evaluation for some parameters in the model by either include it in the model belief base, or even opting it out of the BBFM, in order to satisfy the belief base features requirements.

Definition 5.6. Truth Assumption is a quantified measure for a given parameter, indicating its probability of obtaining satisfiable truth assignment after the reasoning process. If the truth assumption was ultimately equal one, this implies, the parameter is definitely existing in all products configurations.

In addition, belief base could be subjectively defined, such that we assign a truth assignment to some non-core features, to emphasize some model preferences, or obtain some non-functional qualities. The selection of the preferred features is also serving in forming the truth flow of the belief model, which could be identified judiciously to optimize the reasoning process in latter stages.

Belief base is a set of parameters in the belief model, such that each parameter is assigned with truth assignment equal one; either as a part of core features or due to subjective selection.

Having a belief base would improve the belief knowledge, concluding new degree of certainty of other features' truth assumptions.

To quantify the degree of certainty of all models' parameters, we need to identify the truth flow from belief base to all other parameters; while maintaining the existing semantics of the involved dependency contexts.

In the following subsection, we propose the mathematical scheme, enabling us to anticipate the truth assumption of all model parameters.

5.4.1. Computing the Truth Assumption of Model Parameters

Quantifying the truth assumption of any parameter is subject to the feature dependencies, and its association with other parameters.

To thoroughly understand the implication of involved parameters on a given feature, we will study all proposed dependency contexts and introduce the mathematical approach identifying the truth assumptions in each.

Corollary 5.2. Let f be a feature generated via Definition 5.1, with a constraint assignment c_f . We define the truth assumption of f as follow;

$$P(f \mid c_f) = \begin{cases} 0, & c_f = 0 \\ \omega_c, & 0 < c_f < 1 \\ 1, & c_f = 1 \end{cases}$$

Proof and Semantic The truth assumption of any feature is constrained by the truth assumption of its corresponding crosstree constraints. Whereas, if feature f exhibits an exclusion dependency with another feature with truth assignment equal 1. Similarly, its corresponding c_f value equal zero. Consequently, this means feature f must be excluded from the system configurations, and therefore to be assigned zero probability.

In the other scenario whereas $0 < c_f < 1$, the probability of obtaining a truth assignment of feature f would change accordingly as per ω_c, whereas ω_c is determined in accordance with the semantic of the involved dependency context.

Finally, when $c_f = 1$ this implies that feature f is required by a core feature with truth assignment equal one, therefore it should be true in all model configurations; hence it would be assigned truth assignment value equal one.

Definition 5.7. Let *f* be feature in BBFM, *f* has a probabilistic truth assumption ω_f ,whereas ω_f is a measure of the feature dependency's generalization and any crosstree constraints *f* might exhibit.
In addition, ω_f is also determined by the nature of the dependency context, in which it interacts with the adjacent compound feature f_c and any sibling features.

To compute the truth assumption of any feature we would check its dependency context semantic as follows;

5.4.1.1. Bayesian Conjunction

Proposition 5.1. Let x be subfeature generated via definition 5.2, we define ω_x as;

$$\omega_x = P(x \mid f_c, c_s)_{\theta_*} = \begin{cases} 1 & ,P(f_c) = 1 \\ P(f_c)\left[I(c_f)\right] & ,otherwise \end{cases}$$

Proof and Semantic Definition 5.2 requires that all subfeatures must be true in all system configurations wherever its compound feature f_c is true.

Proposition 5.1 meets the semantic of Definition 5.2 in which, if the compound feature f_c has a truth assignment equal 1, this implies that the compound feature f_c is true in all system configurations; therefore all its generalization and specialization must be within a valid dependency context.

To satisfy Definition 5.2 and Lemma 5.1; x must also be true in all system configurations. Therefore, its probabilistic truth assignment must equal one, regardless of any kind of crosstree constraints x might exhibit.

In this context *x* has the highest priority in the model belief system. Similarly, if *x* has an exclusion dependency via crosstree constraint, ultimately the entangled feature must be excluded from all models' configurations.

In the other scenario, if the truth assumption of the adjacent compound feature doesn't qualify f_c to be true in all system configurations, this semantically disqualify *x* of being true in all systems' configuration, whereas its truth assignment is determined by the truth assumption generalization, as per $P(f_c)$.

Unlike the first scenario, here we should consider the implication of all crosstree dependency. In which, its truth assumption might require to exclude *x* from system configuration wherever applied.

To illustrate the implication of crosstree dependencies, we generate dependency array $I(c_f)$; combining all mutual crosstree constraints *x* might exhibit; quantifying the implication of any crosstree constraint on *x*. such that $I(c_f)$ equal the truth assumption of features with requires dependency and truth assumption complement of features with exclusion dependency.

Definition 5.8. Let *E* be a set of subfeatures grouped in Bayesian dependency context with a compound feature *f*.

we denote the dependency function as θ such that, the semantic evaluation of the dependency context θ, could be identified via γ ; where γ is a result of truth assumption of all adjacent subfeatures. γ Is determined by the type of the dependency context of the associated subfeatures and its specialization. Moreover, it also figures out the righteousness truth evaluation in validating the dependency context.

Proposition 5.2. Let x be set of observed subfeatures generated via Definition 5.2; to evaluate truth assumption of the dependency context, we compute γ by recalling Bayes theorem as follow;

$$\gamma_x = P(\theta_\bullet \mid x) = \frac{P(\theta_\bullet)P(x \mid \theta_\bullet)}{P(\theta_\bullet)P(x \mid \theta_\bullet) + P(\overline{\theta_\bullet})P(x \mid \overline{\theta_\bullet})}$$

Proof and Semantic Definition 5.2 and Definition 5.8 imply that, the truth assignment of the conjunction function $\theta_{\bullet}$ is conditioned by the truth assumption of its all its adjacent subfeatures.

Therefore, if we observed set of subfeatures x in within a conjunction dependency context, such that $\{x\} \in \{all\ \ subfeatures\}$,
We can evaluate the truth assumption of $\theta_{\bullet}$ in accordance with its truth assumption and Lemma 5.1.

All features f in x will be evaluated via Corollary 5.2, such that if all features f in the set of observation $\{x\}$, succeeded to have a truth assignment greater than zero $\forall f = \omega_f$ such that $f \in \{x\}$. γ In Lemma 5.2 is satisfied, therefore $\theta_{\bullet}$ will have a truth assignment computed as per Proposition 5.2. γ Is determined by the number of truth observations, and it only can achieve truth assignment equal one, and validate the conjunction context $\theta_{\bullet}$; if all subfeatures were observed with a truth assignment. Likewise, if one or all observed subfeatures were detected with truth assumption equal 0, γ cannot be valid with probabilistic truth assumption equal zero.
To validate $\theta_{\bullet}$, and achieve a truth assignment equal 1, such that γ =1; the second term of denominator must be equal zero, $P(\overline{\theta_{\bullet}})P(x \mid \overline{\theta_{\bullet}}) = 0$.

Since our observation set is $\{x\}$, we will investigate the settings, thus will set this term to be zero *i.e* $P(x \mid \overline{\theta_{\bullet}}) = 0$

To examine this notion, suppose we have a set of four subfeatures grouped in conjunction function, as follow $\theta_{\bullet} : \{f_c, f_{sa} f_{sb} f_{sc} f_{sd}\}$ in which two subfeatures x exhibit a truth assignment; such that $\{x\} = \{f_{sa}, f_{sb}\}$.

$P(x \mid \overline{\theta_{\bullet}})$ Is $\dfrac{P(x \cap \overline{\theta_{\bullet}})}{P(\overline{\theta_{\bullet}})}$, by recalling Definition 5.2 we conclude that $\theta_{\bullet}$ can only be assigned a truth assignment, if the set of all observed subfeatures combined all subfeatures. Hence, only one set can contain all subfeature and assign a truth assignment of the conjunction context.

The probability of obtaining this set is $P(\theta_\bullet)=\frac{1}{2^n}$, alternatively $P(\overline{\theta_\bullet})=\frac{2^n-1}{2^n}$.

Similarly, we can infer that $P(x\cap\overline{\theta_\bullet})=\frac{3}{2^n-1}$. This will not set $P(x\mid\overline{\theta_\bullet})$ to be zero. Only one selection can do so, in which we observe all subfeatures with truth assumption such that, $\{x\}=\{all\ \ subfeatures\}$.

In this case, $P(x\cap\overline{\theta_\bullet})=0$; as this is embedded by Definition 5.2. When $P(x\cap\overline{\theta_\bullet})=0$; the term $P(\overline{\theta_\bullet})P(x\mid\overline{\theta_\bullet})$ is also equal zero. Consequently, we can conclude $\gamma_x=P(\theta_\bullet\mid x)=1$.

This proves our assumption and Proposition 5.2, in which, only one set can assign a truth assignment of the conjunction context, when all subfeatures are observed with a truth assumption greater than zero.

In further discussion, to evaluate the truth assumption γ given that we observed set of features x . Such that,

$$x\in\{i=0,i=1,...,i=n\}$$

i is the number of contained subfeatures in the observation set, and n is the number of all subfeatures.
Problem space Ω is 2^n, only one event can evaluate $\theta_\bullet$ to be true, as we have explained earlier in this section. Therefore, $P(\theta_\bullet)=\frac{1}{2^n}$.
Now to evaluate the truth assignment of $\theta_\bullet$ given *i,* $P(\theta_\bullet\mid i)$*;* we will use Bayes Theorems as follow:

1. Calculate how many subfeatures can be observed *i.* $i\in\{0,1,2,..,n\}$, whereas n is the number of observed subfeatures in each combination.
2. Compute the probability of getting an event contain exactly *i feature,*
   Such that;

$$P(i)=\frac{\begin{pmatrix} n \\ i \end{pmatrix}}{2^n}$$

3. Compute the probability of observing all subfeatures,
   Such that;

$$P(i)=\frac{\begin{pmatrix} n \\ n \end{pmatrix}}{2^n}=\frac{1}{2^n}$$

4. Compute the probability of getting at least *i* count of subfeature in an event combined of *j* subfeatures. $P(i \mid j)$.

   Such that;

$$\begin{aligned} &P(i=0)\cup P(i=1)\cup ..\cup P(i=n) \\ &= \\ &P(i=0)+P(i=1)+..+P(i=n) \\ &=1 \end{aligned}$$

   Similarly, each event *i* has *x* possible combination whereas $x = \binom{n}{i}$ And,

$$\begin{aligned} &\widetilde{P}_x(i=0)\cup \widetilde{P}_x(i=1)\cup ...\cup \widetilde{P}_x(i=n) \\ &= \\ &\widetilde{P}_x(i=0)+\widetilde{P}_x(i=1)+...+\widetilde{P}_x(i=n) \\ &=\frac{\begin{pmatrix} n \\ i \end{pmatrix}}{2^n} \end{aligned}$$

   To normalize the measurement, we divide each term by the total and we get

$$\begin{aligned} &p_x(i=0)+p_x(i=1)+..+p_x(i=n) \\ &=1 \end{aligned}$$

   Knowing we have already observed *j* subfeature, what is the probability to find a combination that at least contain *i* number of subfeatures, in *x* number of events, such that we have *i* number of features in each event. Where is *i* is the number of possible observed subfeature.

$$P(j \cap x_i) = P(j \cap (x_{i=1} \cup x_{i=2} \cup .. \cup x_{i=x}))$$
$$=$$
$$P(j \cap x_{i=1}) \cup P(j \cap x_{i=2}) \cup .. \cup P(j \cap x_{i=x})$$

In case we observed j was bigger than x_i , this means number of observed features is bigger than number of subfeatures contained in x_i

$$j \cap x_i = \emptyset.$$

This means,

$$P(j \cap x_i) = P(\emptyset \cap \emptyset \cap .. \cap \emptyset)$$
$$= 0$$

In case we observed j was equal or less than x_i , this means number of observed features is equal or less than number of subfeatures contained in x_i, $j \cap x_i = x_i$.

Similarly,

$$P(j \cap x_i) = P(x_i \cap x_i) = 1$$

Now giving that we computed all above, we can extend our calculations, to find probability of the conjunction function giving an exact number of observed subfeatures as follow;

$$P(\theta_\bullet \mid j) = \frac{P(j \mid \theta_\bullet)}{P(j)}$$

$$= \frac{P(j \mid \theta_\bullet) P(\theta_\bullet)}{\sum_{i=0}^{n} P(j \mid i) P(i)}$$

Whereas j is number of observed subfeatures and n is the number of all subfeatures.

5.4.1.2. Bayesian Disjunction

Proposition 5.3. Let x be a set of s number of observed subfeature, generated via Definition 5.3 we define ω as;

$$\omega_x = P(x \mid f_c, c_s)_{\theta_+} = \left(\frac{2^{n-s}}{2^n - 1} P(f_c)\left[I(c_s)\right] \right)$$

Proof and Semantic Definition 5.3 demands that at least one subfeature must exhibit a truth assignment, to evaluate the compound feature a truth assignment. The truth assignment of any subfeature is independent from other subfeatures in the same context.

To examine this proposition; assume that we have a set of four equiprobable subfeatures grouped in disjunction function, such that,

$$\theta_+ : \left\{ f_c, f_{sa} f_{sb} f_{sc} f_{sd} \right\}.$$

Using the notation 2^x to denote the set of all subsets of x. We want to define the probability weight of subfeature f_{sd}, $P(f_{sd})$.

Since the subfeature f_{sd} is actually a set of all subsets $E \subset \left\{ f_{sa}, f_{sb}, f_{sc}, f_{sd} \right\}$ such that $f_{sd} \in E$ which is in bijection with f_{sa}, f_{sb} and f_{sc}, $P(f_{sd}) = \frac{\left|2^{\{f_{sa}, f_{sb}, f_{sc}\}}\right|}{\left|2^{\{f_{sa}, f_{sb}, f_{sc}, f_{sd}\}}\right|} = \frac{1}{2}$.

Similarly,

$$P(f_{sb} f_{sd}) = \frac{\left|2^{\{f_{sa}, f_{sc}\}}\right|}{\left|2^{\{f_{sa}, f_{sb}, f_{sc}, f_{sd}\}}\right|} = \frac{1}{4}, \text{ and } P(f_{sd} \mid f_{sc}) = \frac{P(f_d \cap f_c)}{P(f_{sc})} = \frac{1}{2}.$$

These values are a result of the hold independence assumption.

Define the problem space Ω, such that; $\Omega = \left\{ 2^n \right\}$.

Since, Definition 5.3 implies that all subsets evaluate θ_+ to be true except one subset, in which none of the subfeatures is true $\varnothing$; the truth domain of θ_+ is $\Omega - 1$.

Moreover, the probability of any subfeature $P(f)$ is actually number of events in which f is true over the problem space $P(f) = \frac{\#of\ occrances}{2^n}$.

Similarly $\#of\ occrances = 2^n P(f)$

$P(f) = \frac{2^{n-s}}{2^n}$, Where s is number of subfeatures in event f.

As explained above, Now we can conclude that the probability of having s subfeature within true θ_+ , is actually equal number of occurrences of the subfeature over number of valid truth space $\frac{2^{n-s}}{2^n - 1}$.

In addition, we also know that, the truth assumption of x, is determined by its truth generalization, which is $P(f_c)$, alongside with the implication of all corresponding constraints.

Considering all the aforementioned measurements, we can now compute ω as per hold in Proposition 5.2.

Proposition 5.4. Let x be set of observed subfeatures generated via Definition 5.3. To find out the truth assumption of the dependency context γ we recall Bayes theorem as follows;

$$\gamma_x = P(\theta_+ \mid x) = \frac{P(\theta_+)P(x \mid \theta_+)}{P(\theta_+)P(x \mid \theta_+) + P(\overline{\theta_+})P(x \mid \overline{\theta_+})}$$

Proof and Semantic Definition 5.3 and Proposition 5.3 imply that, the truth assumption of the disjunction function θ_+ is conditioned by the truth assumption of its succeeded adjacent subfeatures.

Therefore, if we observed set of subfeatures x in within a disjunction context such that,

$$\{x\} \in \{all\ subfeatures\}$$

we can conclude the truth assignments of θ_+ in accordance with our observation and Lemma 5.2.

All observed features f will be evaluated via Corollary 5.2, whereas if one feature f in the observation set $\{x\}$ succeeded to have a truth assignment greater than zero $\exists f = \omega_f$ such that $f \in \{x\}$; Lemma 5.2. is satisfied, therefore θ_+ will be qualified to have a truth assignment computed as per Proposition 5.3. γ is determined by the truth observations of the involved subfeatures, and it only can achieve truth assignment equal 1 and activate the disjunction function θ_+ , when at least one subfeatures exhibit a truth assignment. Likewise, if none of the subfeatures exhibits a truth assumption, γ would be weighed a probabilistic truth assumption equal 0.

To activate θ_+ and obtain a truth assignment such that γ =1, the second term of denominator must be equal zero $P\left(\overline{\theta_+}\right)P\left(x \mid \overline{\theta_+}\right) = 0$.

To examine this notion, assume that we have a set of four equiprobable subfeatures, grouped in disjunction function, such that,

$$\theta_+ : \{f_c, f_{sa} f_{sb} f_{sc} f_{sd}\}$$

We observed set x, such that x contain at least one subfeature, probability of $P\left(x \mid \overline{\theta_+}\right)$, will be $\dfrac{P\left(x \cap \overline{\theta_+}\right)}{P\left(\overline{\theta_+}\right)}$.

Following Definition 5.3, θ_+ can only be true if at least one subfeature exhibit a truth assignment, therefore all sets that contain at least one true subfeature can validate the disjunction function, with probability equal;

$$P(\theta_+) = \frac{\sum_{i=1}^{n} \frac{n^i}{i!}}{2^n}.$$

Having that said, $x \cap \overline{\theta_+} = \varnothing$, whereas x has at least one subfeature with truth assignment.

Consequently, $P(\varnothing) = 0$, which will set the second term of the dominator to zero. Thus, proves the semantic of our assumption and Proposition 5.4, which suggests that if we observe at least one subfeature with truth assignment; disjunction function would be assigned a valid truth assignment.

In further discussion, to compute the probability of obtaining x set of subfeature, such that x consist of s number of subfeatures $P(s \mid \theta_+)$ and $s \geq 1$.

we recall conditional probability law as follows;

$$P(\theta_+ \mid s) = \frac{P(\theta_+)P(s \mid \theta_+)}{P(s)}$$

Alternatively,

$$P(s \mid \theta_+) = \frac{P(s)P(\theta_+ \mid s)}{P(\theta_+)}$$

Knowing that, $P(\theta_+ \mid s) = 1$ and probability of obtaining s $P(s) = \frac{\frac{n^s}{s!}}{2^n}$, we can conclude that;

$$P(s \mid \theta_+) = \frac{\frac{n^s}{s!}}{\sum_{i=1}^{n} \frac{n^i}{i!}} \quad , s \neq 0$$

Let $\theta_+(f_c, f_1 .. f_4)$ be a Bayesian Disjunction dependency context such that, four subfeatures are grouped in Bayesian disjunction dependency, with a compound feature.

Giving that the dependency context is satisfied with a valid truth assignment, Figure 5. 10 shows the probability of obtaining s number of valid features, as per computed above.

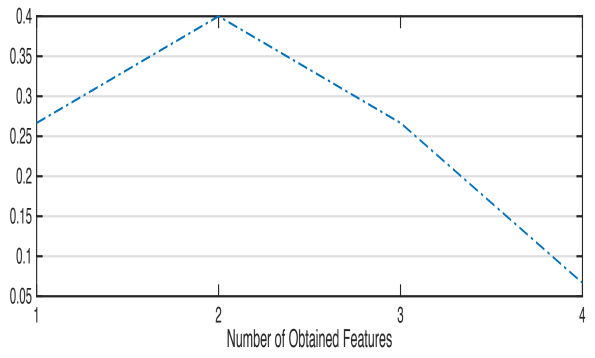


Figure 5. 10 probability of obtaining s number of valid features in valid Bayesian disjunction

5.4.1.3. Bayesian Exclusive Disjunction

Proposition 5.5. Let x be a set of s observed subfeatures generated via Definition 5.4, we define ω as;

$$\omega_f = P(x \mid f_c, c_s)_{\theta_\oplus} = \left(\frac{1}{n} P(f_c) \left[I(c_s) \right] \right)$$

Proof And Semantic Definition 5.4 demands that, only one subfeature must be true to evaluate the compound feature a truth assignment. The truth assumption of any subfeature in valid exclusive disjunction function could be dependent on the truth assumption of other subfeatures in the same context.

To examine this proposition, assume that we have a set of four equiprobable subfeatures, grouped in exclusive disjunction function, such that,

$$\theta_\oplus : \{ f_c, f_{sa} f_{sb} f_{sc} f_{sd} \}$$

Using the notation 2^x, to denote the set of all subsets of x. For instance, we want to define the probability weight of subfeature f_{sd}, $P\left(f_{sd}\right)$. The problem space Ω is equal 2^n, only events with one valid subfeature will validate $\theta_{\oplus}$ semantic.

We use binomial coefficient with falling factorial notation $\frac{n^{\underline{i}}}{i!}$ to find out how many events in the problem space contain only one subfeature. i is number of subfeatures in each events; in our favor we need only one subfeature to appear in each event, to meet exclusive disjunction semantic, such that $i=1$.

Consequently, only n events will consists only of one subfeature. Thus, form the exclusive disjunction truth domain. The probability of obtaining a given subfeature in a valid $\theta_{\oplus}$ is $\frac{1}{n}$. Such that, n is the number of subfeatres.

From above discussion we proved that Proposition 5.4 holds the semantic of Bayesian exclusive Disjunction.

To understand the nature of dependency between subfeatures in different scenarios, assume that truth assumption of $\theta_{\oplus}$ is still unknown. In this case, the truth assumption of involved subfeatures are independent of each other such that, $P\left(f_s\right)=\frac{1}{2}$ and the truth assignment of any subfeature doesn't affect the truth assignment of another subfeature $P\left(f_{s1} \mid f_{sn}\right)=P\left(f_{s1}\right)=\frac{1}{2}$.

When we observe that $\theta_{\oplus}$ is valid dependency function, such that,

$$P\left(f_s\right)_{\theta_{\oplus}}=\omega_f=\frac{1}{n}, \text{ and, } P\left(f_{s1} \mid f_{s2}\right)_{\theta_{\oplus}}=\frac{P\left(f_{s1} \cap f_{s2}\right)}{P\left(f_{s2}\right)} \text{ , where, } \left(f_{s1} \cap f_{s2}\right)=\varnothing .$$

Consequently, $P\left(f_{s1} \mid f_{s2}\right)_{\theta_{\oplus}}=0$, which implies that if we observed a subfeature within a valid Bayesian exclusive disjunction context; these infer a mutual exclusion all other sibling subfeatures and set its truth assumption to zero, which semantically hold in Definition 5.4 and Proposition 5.4.

Proposition 5.6. Let f be an observed subfeature generated via definition 3.4 To find out the truth assignment of γ we recall Bayes theorem as follows;

$$\gamma_f = P(\theta_\oplus \mid f) = \frac{P(\theta_\oplus)P(f \mid \theta_\oplus)}{P(\theta_\oplus)P(f \mid \theta_\oplus) + P(\overline{\theta_\oplus})P(f \mid \overline{\theta_\oplus})}$$

Proof and Semantic Definition 5.4 and Proposition 5.5 imply that, the truth assumption of the exclusive disjunction function $\theta_\oplus$ is conditioned by the truth assumption of its follow adjacent subfeatures.

Therefore if we observed an event x in within an exclusive disjunction function, such that, $\{x\} \in \{all\ \ subfeatures\}$ we can evaluate the truth assignments of $\theta_\oplus$, in accordance with our observation and Lemma 5.3.

Features f will be evaluated via Corollary 5.2 whereas if only one feature f in the observation set $\{x\}$, succeeded to have a truth assignment greater than zero, such that all other subfeatures were assigned a zero truth assumption.

Lemma 5.3 is satisfied, therefore $\theta_\oplus$ will have a truth assignment computed as per Proposition 5.6. γ is determined by the truth observations, and it only can achieve truth assignment equal one, consequently, validate the exclusive disjunction function $\theta_\oplus$; if and only if one subfeature were observed with a truth assignment.

By definition, if more than one observed subfeatures were observed with truth assignment equal 1; $\theta_\oplus$ cannot be valid with truth assumption γ equal 0.

To activate $\theta_\oplus$, and achieve the required truth assignment, in which γ = 1, the second term of denominator must be equal zero, $P(\overline{\theta_\oplus})P(x \mid \overline{\theta_\oplus}) = 0$.

Since our observation is around the subfeatures $\{x\}$, we will study how our observation can set this term to zero *i.e* $P(x \mid \overline{\theta_\oplus}) = 0$.

Firstly, we will highlight the semantic of this term, coupled with truth set of $P(x \mid \overline{\theta_\oplus})$.

To answer this, assume we have a set of subfeatures *E*, grouped in an exclusive disjunction function such that $E=\{f_{sa},f_{sb},f_{sc},f_{sd}\}$,

Using the notation 2^E ,to denote the set of all subsets of the set *E*, we have $\Omega=\{2^E\}$. Events with only one subfeature will qualify $\theta_{\oplus}$ a truth assignment such that $\{x\}=\frac{E^{\underline{i}}}{i!}$ whereas $i=1$.

As a result, we can see only E out of 2^E events satisfy Definition 5.4.

$$P(\theta_{\oplus})=\frac{E}{2^E}$$

Alternatively, all other observations disqualify the Bayesian exclusive disjunction of being valid. All other observations are;

$$\{x\}=\sum_{i=0}^{E}\left(\frac{E^{\underline{i}}}{i!}\right)-\left(\frac{E^{\underline{1}}}{1!}\right)=2^E-E\,.$$

Similarly,

$$P(\overline{\theta_{\oplus}})=\frac{2^E-E}{2^E}\,.$$

Noticeably, to obtain $P(\overline{\theta_{\oplus}})$ we exclude all events that has only one subfeature in the event, as per $\left(\frac{E^{\underline{1}}}{1!}\right)$.

Thus, $(x\cap\overline{\theta_{\oplus}})\,suchthat\ \{x_{i=1}\}is\ \varnothing$. Therefore, $P(x\,|\,\overline{\theta_{\oplus}})\,suchthat\ \{x_{i=1}\}is\ 0$.

To conclude, t we can only achieve a valid exclusive disjunction function, such that $\gamma=1$. If our observation set x contained only one subfeature and nothing else, which is also, prove the semantic of Proposition 5.6.

Now, if we observed a certain subfeature with valid truth assignment (due to subjective assignment or dependency requirement), to what extent this would affect our belief about $\theta_{\oplus}\,i.e\ P(\theta_{\oplus}\,|\,f)$!

Following proposition 5.6, γ_f is determined by the truth assignment of $P(\theta_\oplus), P(\overline{\theta_\oplus}), P(f \mid \theta_\oplus)$ *and* $P(f \mid \overline{\theta_\oplus})$.

Previously, we defined $P(\theta_\oplus)$ and $P(\overline{\theta_\oplus})$. To compute the $P(f \mid \theta_\oplus)$ we recall conditional probability law as follow ,

$$P(f \mid \theta_\oplus) = \frac{P(f \cap \theta_\oplus)}{P(\theta_\oplus)}.$$

According to Proposition 5.5, we can conclude that $P(f \mid \theta_\oplus) = \frac{1}{E}$ whereas E is number of involved subfeatures. In addition, our analysis above had shown that $P(\theta_\oplus) = \frac{E}{2^E}$.

Consequently, $\frac{P(f \cap \theta_\oplus)}{\frac{E}{2^E}} = \frac{1}{E}$, and $P(f \cap \theta_\oplus) = \frac{1}{2^E}$.

5.4.1.4. *Bayesian Tautology Context*

Proposition 5.7. Let x be a set of s observed subfeature generated via Definition 5.5, We define ω as;

$$\omega_x = P(x \mid f_c, c_s)_{\theta_T} = \left(\frac{1}{2} P(f_c)\left[I(c_s)\right]\right)$$

Proof And Semantic Definition 5.5 imply that the compound feature is qualified a truth assumption, Regardless of the truth assignment of the adjacent subfeatures. The truth assignment of any subfeature, is independent from other subfeatures in the same context.

Therefore, all subfeatures typically have the same probabilistic truth assignment (when no crosstree constraints are involved). In addition, observing a truth value of compound feature, doesn't necessary mean any of its subfeatures is true.

To examine this proposition, we will assume that we have a set of four equiprobable subfeatures grouped in Tautology function, such that $\theta_{\mathrm{T}} : \{ f_c , f_{sa} f_{sb} f_{sc} f_{sd} \}$.

Using the notation 2^x to denote the set of all subsets of x.

The semantic of tautology context would be satisfied under any possible assignment. Subfeatures are also independent of each other, and there's absolutely no dependency assumption between them (assuming they don't exhibit any mutual dependency among them).

To define the probability weight of subfeature f_{sd}, $P(f_{sd})$. Such that f_{sd}, is actually a set of all subsets $E \subset \{ f_{sa} , f_{sb} , f_{sc} , f_{sd} \}$,

Similarly, $f_{sd} \subset E$. which is in bijection with f_{sa} , f_{sb} and f_{sc}. $P(f_{sd}) = \frac{\left| 2^{\{ f_{sa} , f_{sb} , f_{sc} \}} \right|}{\left| 2^{\{ f_{sa} , f_{sb} , f_{sc} , f_{sd} \}} \right|} = \frac{1}{2}$.

Similarly, $P(f_{sb} f_{sd}) = \frac{\left| 2^{\{ f_{sa} , f_{sc} \}} \right|}{\left| 2^{\{ f_{sa} , f_{sb} , f_{sc} , f_{sd} \}} \right|} = \frac{1}{4}$ and $P(f_{sd} \mid f_{sc}) = \frac{P(f_d \cap f_c)}{P(f_{sc})} = \frac{1}{2}$.

Which concludes that; The probability of any subfeature equal $\frac{1}{2}$.

In addition, we also know that, the truth assignment of x is determined by its generalization truth assignment, which is, $P(f_c)$ and the implication of all involved crosstree constraints.

The above analysis proves the semantic of Proposition 5.7.

Proposition 5.8. Let x be set of observed subfeatures generated via Definition 5.5. ,To find out the truth assignment of γ, we recall Bayes theorem as follows,

$$\gamma_x = P(\theta_{\mathrm{T}} \mid x) = \frac{P(\theta_{\mathrm{T}}) P(x \mid \theta_{\mathrm{T}})}{P(\theta_{\mathrm{T}}) P(x \mid \theta_{\mathrm{T}}) + P(\overline{\theta_{\mathrm{T}}}) P(x \mid \overline{\theta_{\mathrm{T}}})} = 1$$

Proof and Semantic Definition 5.5 and Proposition 5.5 imply that, the truth assignment of the tautology function $\theta_{\top}$ is true among all problem space.

$$P(\theta_{\top}) = \frac{\sum_{i=0}^{n} \frac{n^{i}}{i!}}{2^{n}} = 1$$

Consequently, $P(\overline{\theta_{\top}}) = 0$ which evaluate γ_x to be always equal 1.

To examine this proposition, we will assume that we have a set of four equiprobable subfeatures grouped in tautology function, such that;

$$\theta_{\top} : \{ f_c, f_{sa} f_{sb} f_{sc} f_{sd} \}.$$

When observing x, such that $x \in \{ f_{sa}, f_{sb}, f_{sc}, f_{sd} \}$. $P(x \mid \theta_{\top})$ is $\frac{P(x \cap \theta_{\top})}{P(\theta_{\top})}$ following the Definition 5.5 $\theta_{\top}$ is always true.

We recall Bayes theorem as follow;

$$P(x \mid \theta_{\top}) = \frac{P(x) P(\theta_{\top} \mid x)}{P(x) P(\theta_{\top} \mid x) + P(\overline{x}) P(\theta_{\top} \mid \overline{x})}$$

Whereas, $P(x)$ is already computed via Proposition 5.6 and equal $\frac{1}{2}$ and $P(\theta_{\top} \mid x)$ is equal $P(\theta_{\top})$ equal 1.

By substituting these values, we conclude that $P(x \mid \theta_{\top}) = \frac{1}{2}$.

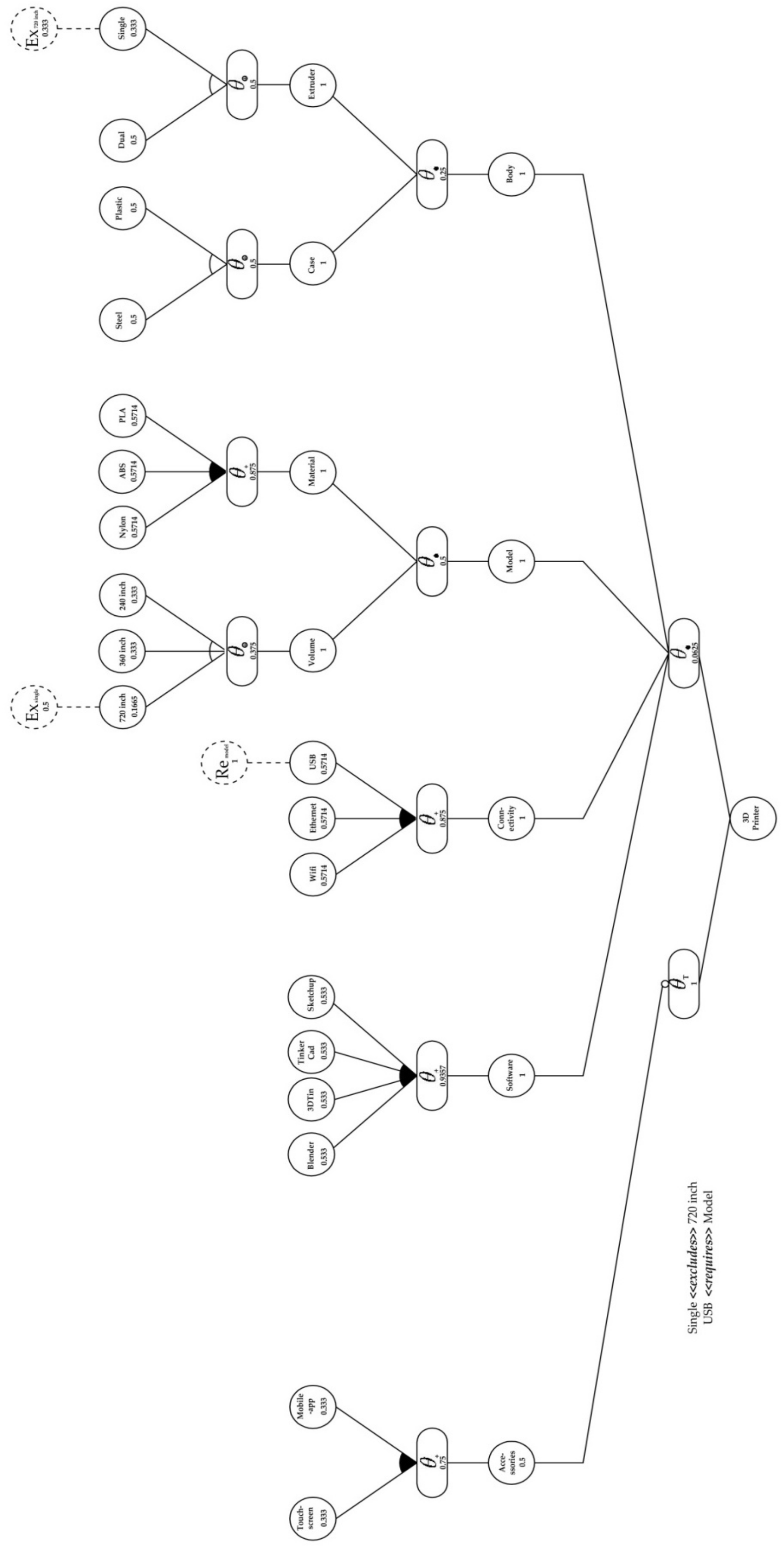


Figure 5. 11 Translation from 3D Printer FM into BBFM. The truth assumption of each parameter is numerically embedded.

5.5. Extended BBFM and Belief Intensification

Employing conditional probability law and Bayes theorem, among other probability techniques; allowed us to develop Bayesian Belief Feature Model BBFM.

Truth assumptions of any parameter in BBFM could be computed using the proposed mathematical notation. Parameters' truth assumption is an indication of the parameter implications and level of entanglement with other parameters in the same model.

While obtaining the truth assumption of model parameters, the model degree of belief starts mounting. Features' truth assumption served in identifying the level of dependency among features, and the probability of obtaining the corresponding feature in valid product configuration. Whereas, the truth assumption of dependency contexts, aided to drive the dependency semantic of the involved features. Moreover, It framed the degree of belief of the dependency context throughout anticipating its satisfiability probability after reasoning.

Apparently, computing the truth assumption of model parameters stems the model belief, which consequently, provides better understanding of the models' behavior and interaction flow among the model parameters.

We argue that, giving a probabilistic truth weight of each parameter will aid in the satisfiability process while highlighting the features actual influence.

By quantifying the truth assumption of any feature, we allow the model designer to adjust the model systematically to obtain certain functionality, or improve the reasoning time through a change in the model belief, and subjectively intensify or subside some parameters truth assumption; while maintaining the dependency contexts legal semantic.

In addition, obtaining truth assumptions of system parameters will improve the model expressiveness, by emphasizing the truth flow throughout the model parameters.

5.5.1. Physics of Visual Expressiveness in BBFM

Visual expressiveness is distinguished by computing the number of visual variables present in a representation. With conviction, colour tends to be the highest in efficiency when it visual variables are considered. As evidenced through numerous studies, human sight proved to be is highly sensitive, responsive and perceptive to the variation in colour. The human eye captures and recognizes any variation in colour faster and more accurate than distinguishing between shapes. For instance, the variations in colour are identified three times faster than variation in shapes. Moreover, colour tends to be more appealing and easier to remember (Moody, 2009).

Using the predefined probabilistic weights as a benchmark for colouring BBFM; is likely to improve the cognitive understanding of the model truth belief.

After computing the truth assumption of features in BBFM, we will translate the probabilistic weight of each feature into “shades of gray” colouring scheme.

Twelve shades are used to illustrate the intensity of the parameter truth assumption. Parameters with higher truth assumptions’ value would be assigned more shades, than ones with lower truth assumption.

Moreover, a parameter with white colour is an invalid parameter, such that its truth assumptions’ value equal zero. Therefore it won’t be satisfied, neither exist in any possible product configuration.
Whereas, parameters with black colour are core parameters, such that its truth assumption is equal one. This implies that it must exist in all possible products configurations.

To give better understanding of the proposed colouring technique; we will incorporate the colouring scheme into the 3D printer exemplar used throughout the whole dissertation. Figure 5. 12, shows a coloured version of Figure 5. 11.

Noticeably, features implication on the model belief is cognitively easily recognized than before, where colours weren’t introduced to the model.
In addition, to understand the dependency flow in the belief model; we can easily trace the colour change, as per shown below. Semantically, features with higher truth assumptions require more colour, and the opposite hold.

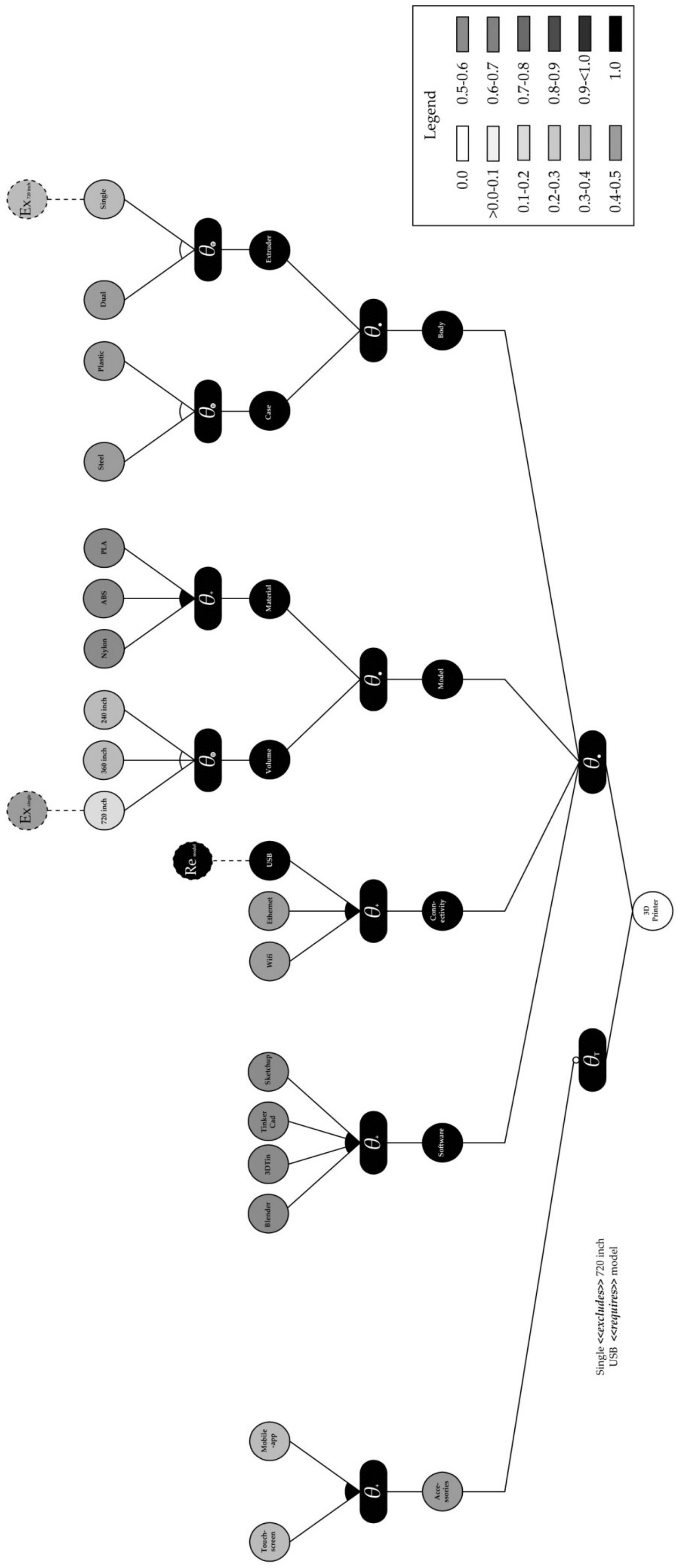


Figure 5. 12 Translation from 3D Printer FM into BBFM. The truth assumption of each parameter is denoted with equivalent shade of gray.

5.5.2. Use of “colour variation” to project the “belief variation”

Figure 5. 13 represents a random BBFM with relatively higher degree of variability. When colouring this model, we enable fast and effective recognition of the model belief assumptions.

In the initial condition as per Figure 5. 13, not much information are given regarding the model’s validation. Thus the satisfiability of the model is still unidentified.

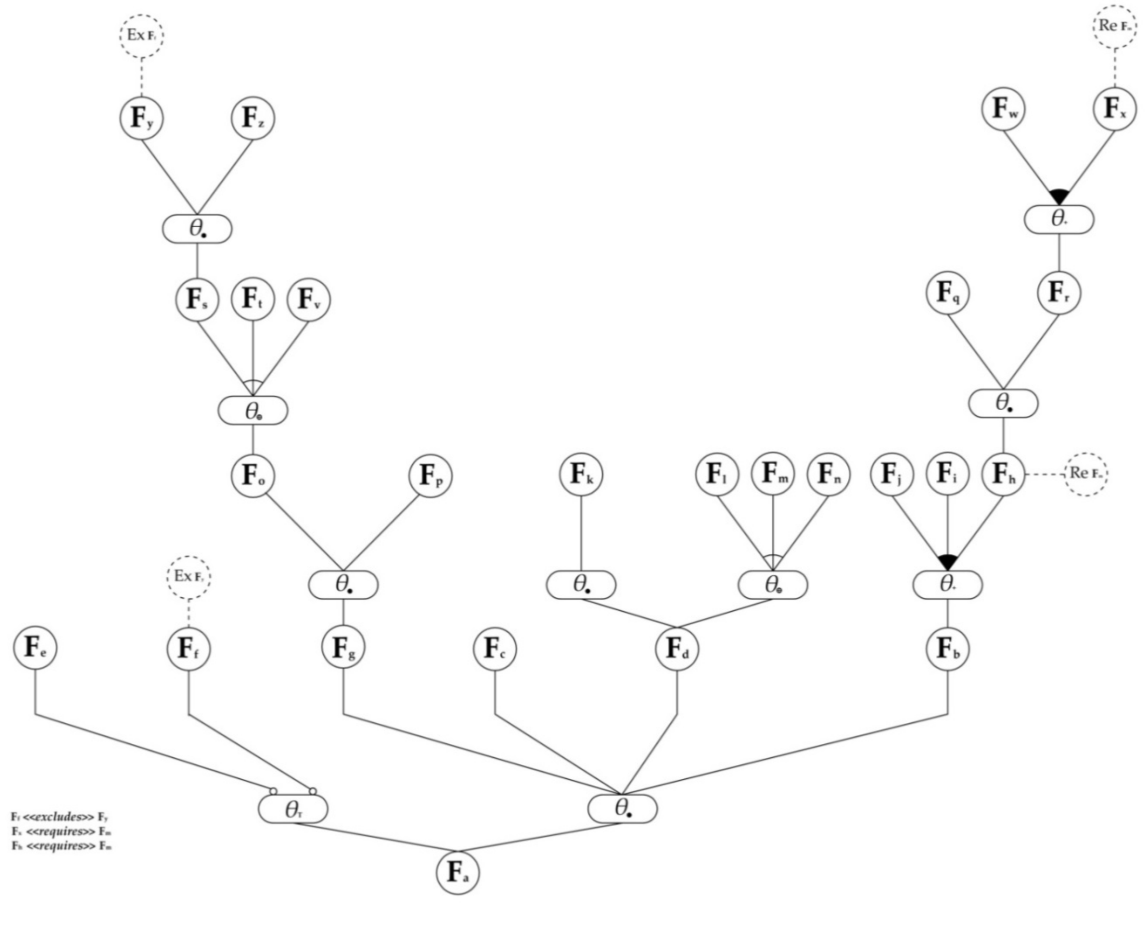


Figure 5. 13 graphical representation of Bayesian Belief Feature Model. , before defining the uncertainty measure of model paramters

Figure 5. 14 below, demonstrates the probability of meeting the intended semantic of each all dependency contexts, in which the probability of having satisfiable dependency context is demonstrated using the colouring method.

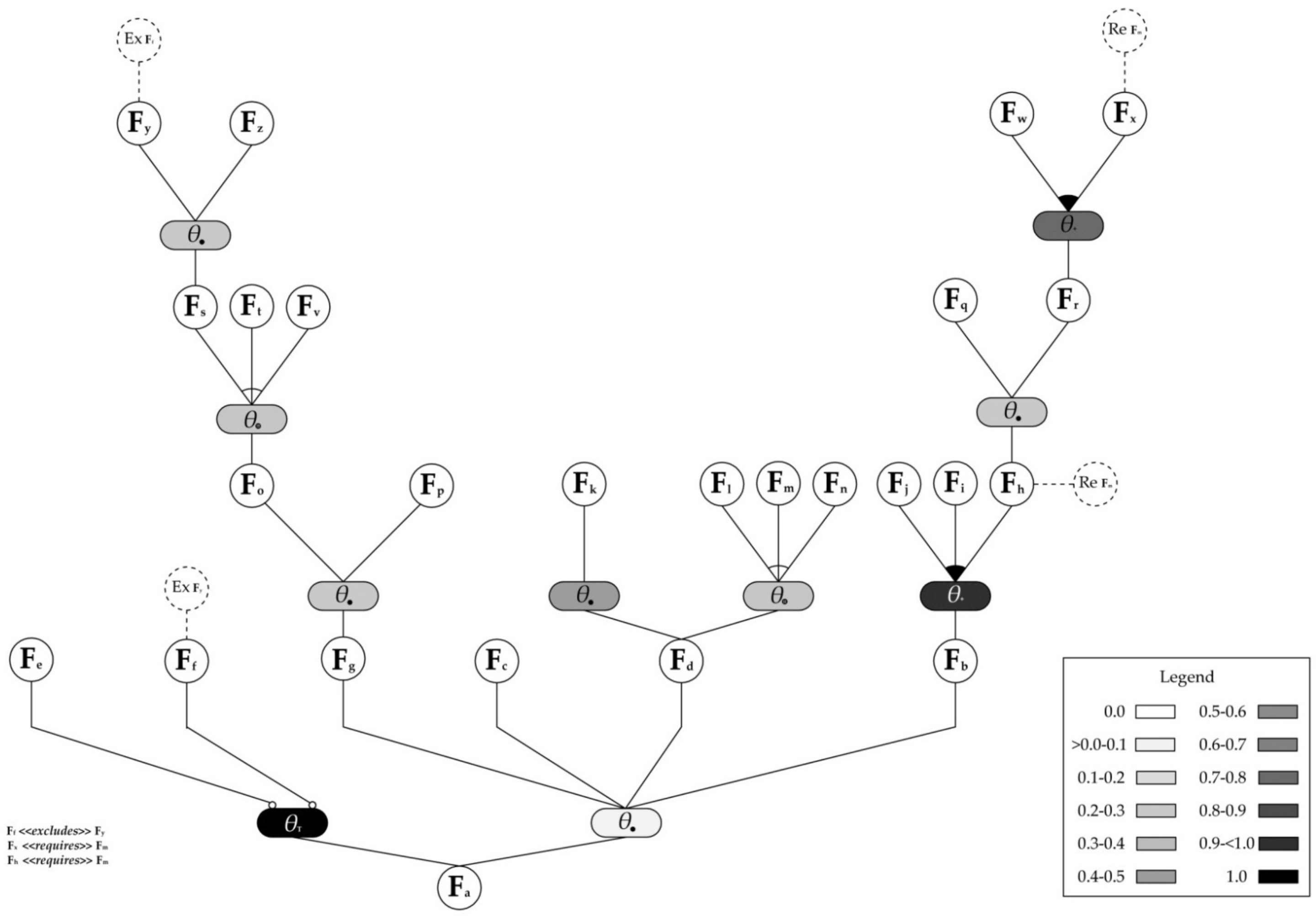


Figure 5. 14 graphical anticipation with "use of colour" for dependency contexts' truth assumption in BBFM

When reasoning, at least all core features must be included in any valid configuration. To satisfy this notion, a flow of change on the model belief would be enforced, and thus translated "colour variation" on both; dependency contexts and features.

Figure 5. 15 illustrate these changes;

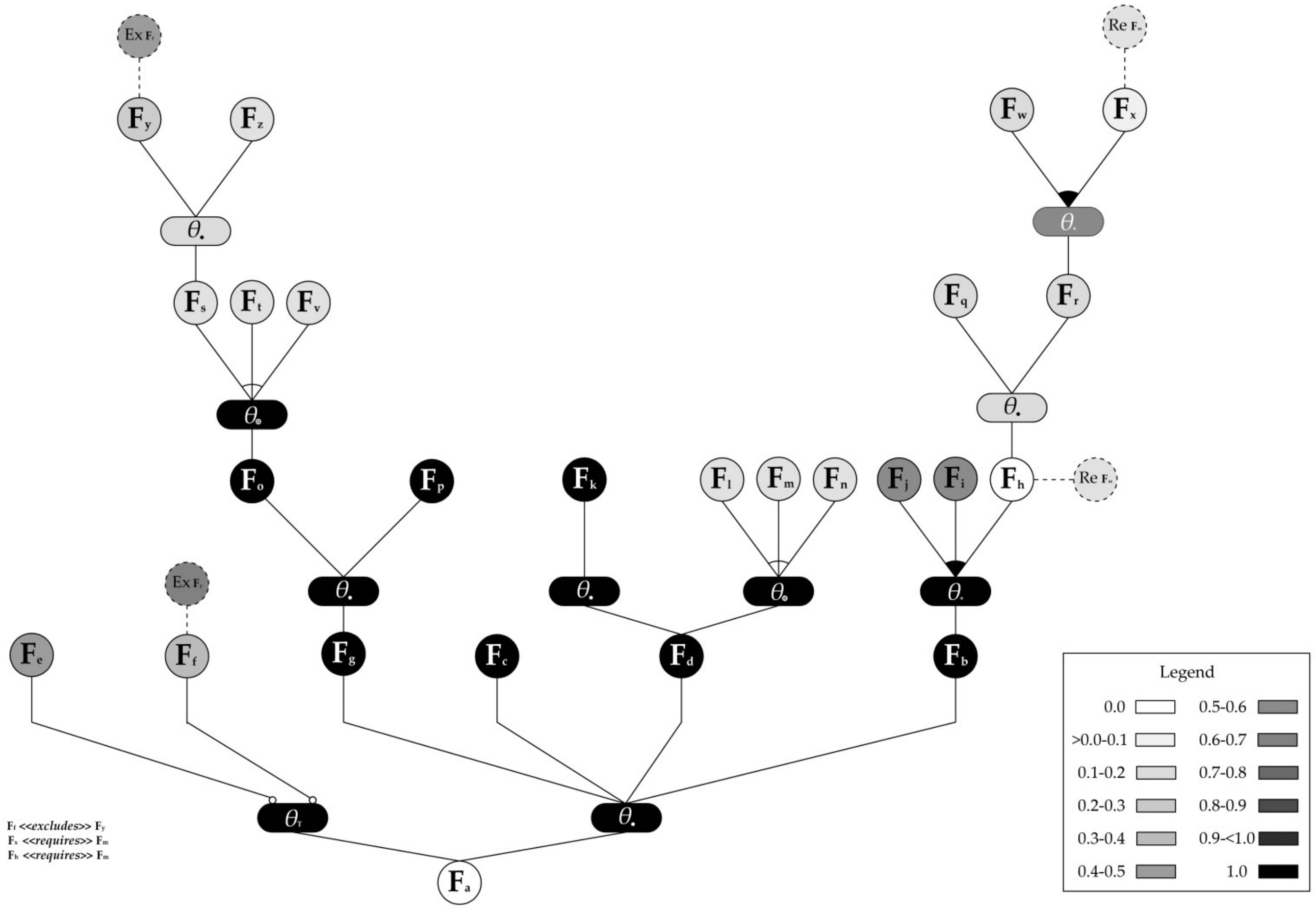


Figure 5. 15 Colour use in BBFM. Different shades of gray are used to denote different truth weights.

Any proposed change on the model belief might be proposed due to different factors. Change might occur due to:

1. The system engineer or the stakeholder decided to include a new feature to the belief base due to (evidentially reason about it) due to:
    - Functionality: It's a required or preferred feature for the targeted functionality or product configuration.
    - Design improvement: It has high probability weight with high dependency influence on other features.

      Therefore, including it in the belief base could reduce the degree of variability of the model (reduce the problem space while enhancing the reasoning process), while maintaining almost the same possible outcomes
    - Its convenience it terms of Availability, whereas other features are less available.

2. The system engineer or the stakeholder decided to remove one of the variable features due to:
    - It's no longer required in most configurations; it has low probability assignment or even zero truth assumption.
    - The features are overpriced or expensive.
    - The features are inaccessible or unavailable.
    - Efficient reasoning: throughout reducing the model complexity.

In the given example, if we decided to include features f_m, f_w in the model belief base, by assigning truth-value equal 1 for both features, a flow of changes would result as per shown in

Figure 5. 16.

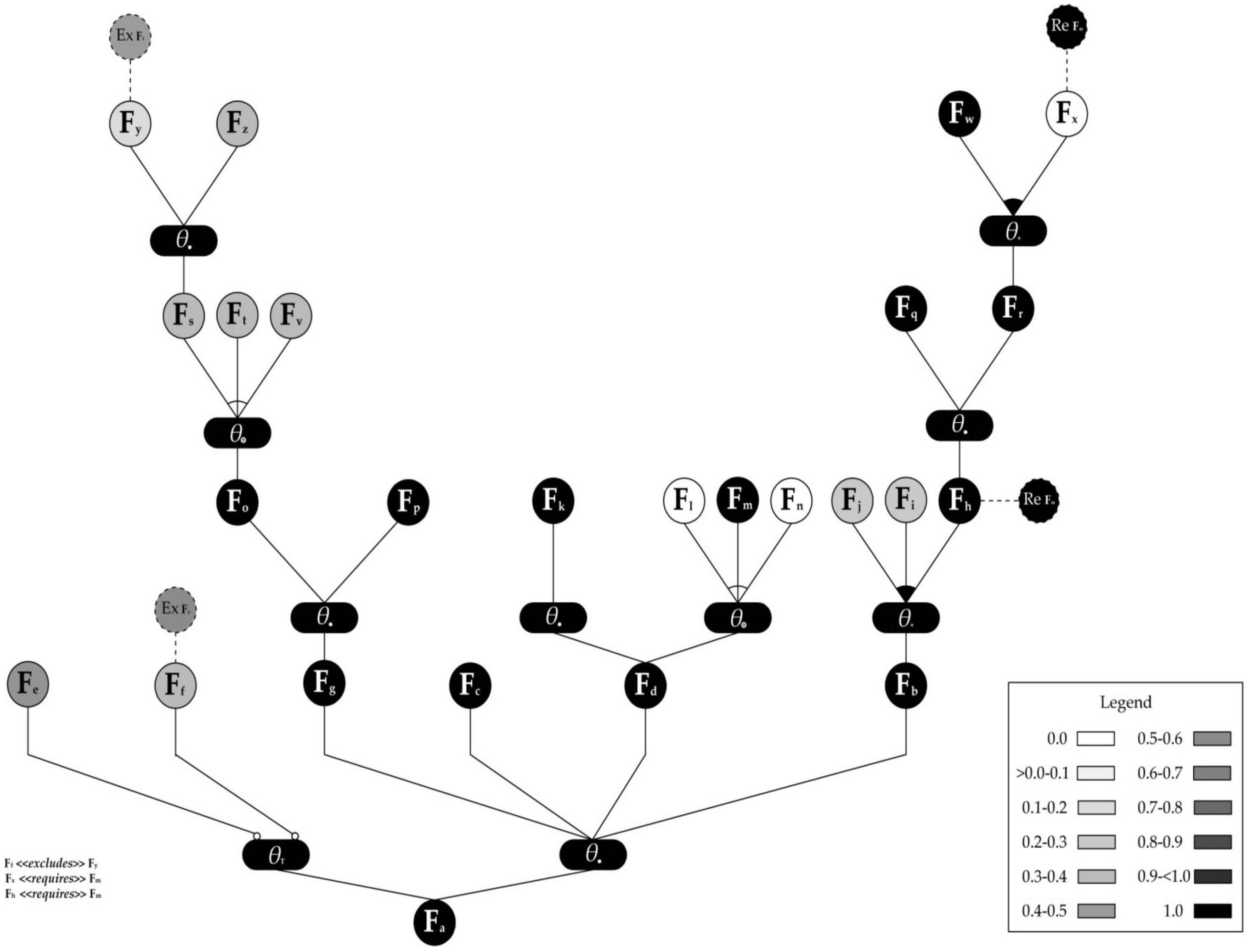


Figure 5. 16 Colour change in figure 5.16, when including features f_m, f_w in the BBFM belief base

In the same fashion, we now decide to include features f_y, f_n in the model belief base. Figure 5. 17 illustrates the new "colour variations" of the model belief assumptions.

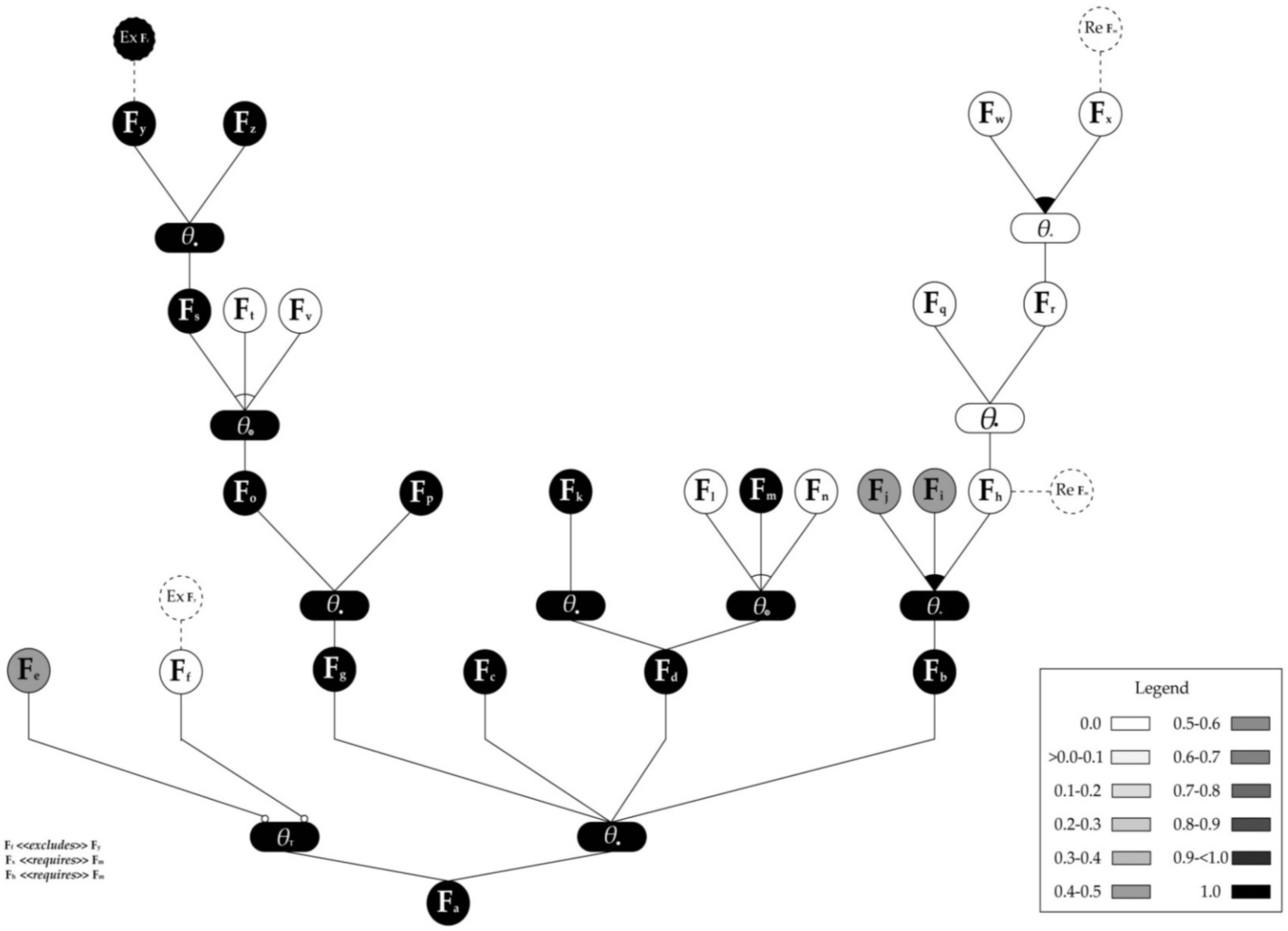


Figure 5. 17 Colour change in figure 5.16, when including features f_y, f_n in the BBFM belief base

Finally, in Figure 5. 18 we decided to include feature f_f in the model belief base.

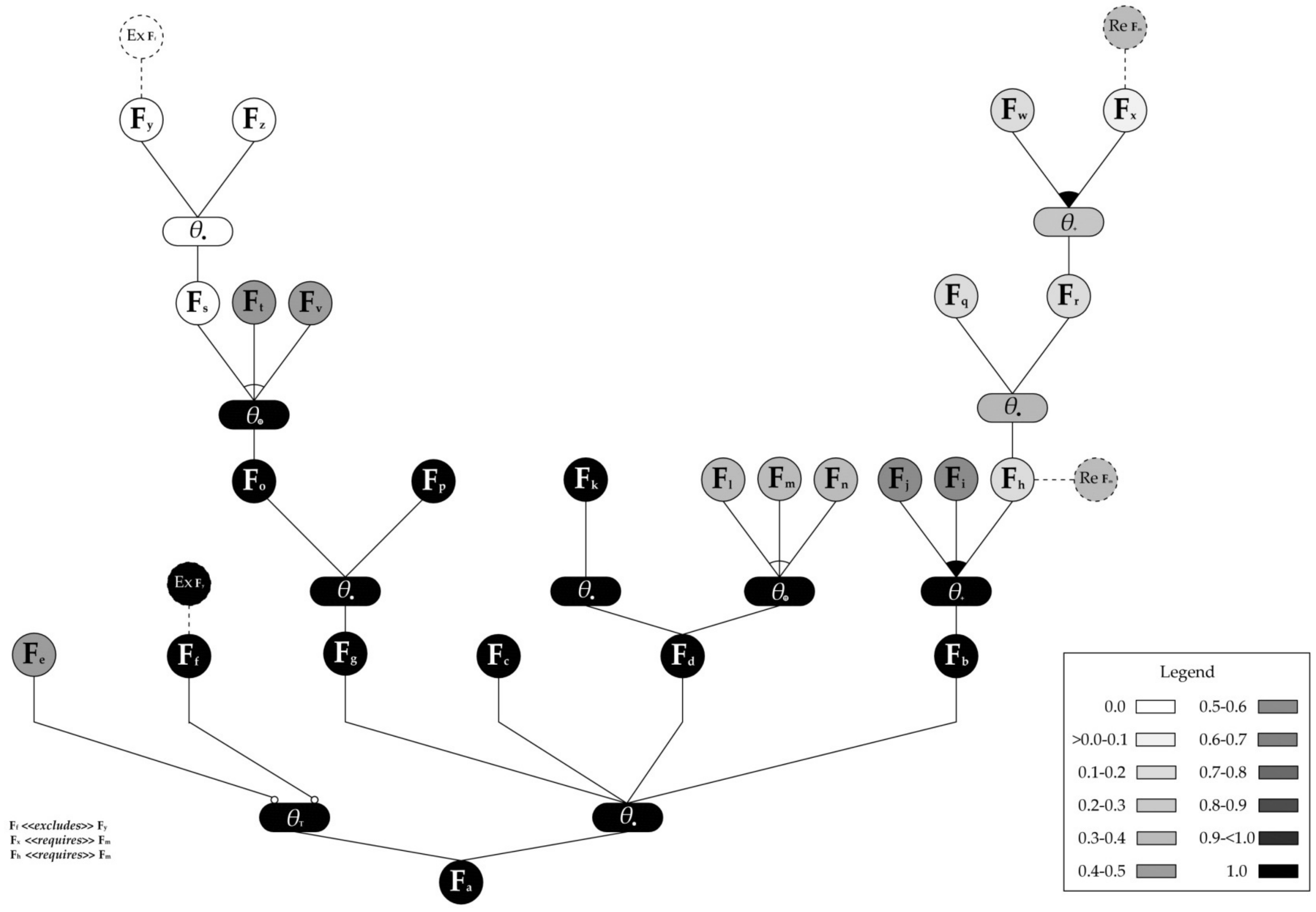


Figure 5. 18 Colour change in figure 5.16, when including features f_f in the BBFM belief base

5.6. Further Discussion and Applicability

In this chapter we thoroughly discussed the mathematical semantic of our developed BBFM, in which we were able to quantify the uncertainty measure of all model parameters. As argued previously, quantifying the uncertainty measure of model parameter would provide a sound estimation of the model behavior, and estimate the actual implication of all model parameters. Instead of assuming the same weight distribution among model parameter, we now can obtain different probabilistic weight for each parameter according to its dependency interaction throughout the model parameters. This lone notion could derive many interesting applications for the developed model.

By exploiting the pre-computed uncertainty measure, we would be able to perceive the level of entanglement between any two features in the model. This enables us to detect the latent non-functional interaction between these two features. Likewise, with every attempt to alter any model by removing or incorporating new features into an existing model, the implication of the new alteration can be detected easily. This grades BBFM as a desirable model for dynamic and growing industries, in which the need of incorporating and excluding some features is a constant need, such as electronics and integrated circuits manufacturing, wherein features are constantly added or removed from the original design.

The pre-computed uncertainty measures could also be used to enhance the physics of the graphical representation of any model, by means of providing an advanced visualization, benefiting from the uncertainty measure variation. Section 5.5 proposes a new simple technique that can be deployed in any model to enhance its visual expressiveness.

In addition, we can benefit from the obtained measure by providing an estimation of the average cost of any product line, through multiplying the likelihood of obtaining any feature and the cost of implementing this feature.

Moreover, BBFM can be integrated into any Variability Management tool to support the decision-making process when configuring all possible products. By defining the dependency flow and the uncertainty measure variation throughout the model, this allow us to decide which parameters need to be included in the desired product, not to only achieve the anticipated functionalities but also to obtain some non tangible functionalities as per stability, complexity, redundancy or simplicity.

In chapter 6 we introduce one major application of the developed model, in which we provide a systematic approach to satisfy the embedded constraints in any model by exploiting the uncertainty measures of model parameters and utilize it to conclude the most apt alternatives aiding in constraints satisfaction problem.

BBFM is based on the traditional notation of feature model; therefore any existing feature model can be easily translated into BBFM using the provided mathematical framework. Having that said, we can argue that BBFM has unrestricted potentials on where to be employed in software product line real life applications. Such as, automobile car manufacturing, diesel engines, electronics, avionics, military technology, aerospace engineering, embedded systems and many more.

Although the focus of this dissertation was on software product line, BBFM can also be used to model any system at which the need of decisions support and truth quantification is a key importance. Therefore, this work can be extended and applied extensively in machine learning, artificial intelligence, robotics, automated decisions and diagnostics models (medical and troubleshooting).

BBFM raises high potentials in term of applicability on various industries, yet the lack of automated computation remain a key challenge specially on large scale systems, at which extensive analysis and computation need to be conducted to quantify the uncertainty measure of the model parameters.

Although in chapter 6 we provide different techniques to ease the computations, modeling on large-scale problem might be costly and difficult. This challenge is to be tackled as a future work and extension of this dissertation.

5.7. Summary

In this chapter, we introduced Bayesian Belief Feature Model BBFM, and thoroughly discussed the mathematical semantic behind it.

Firstly, we started by defining the notion of BBFM and the importance of capturing the uncertainty nature of model parameters.

Having that said, we derived a set of mathematical notations; quantifying the uncertainty measure of the model parameters. Any obtained measure is subject to the model embedded belief. Model's belief stems out of the existing semantics of the introduced dependency contexts, therefore different dependency context allow different truth domain.

To evaluate the truth assumption of any parameter, we need to evaluate its dependency flow. Hence, we presented set of theorems to compute the probability weight of any feature in different dependency contexts. Moreover, we proved the semantic validity of the introduced dependency contexts using Bayes theorem and other probability techniques. By investigating the semantic of the used contexts, we were able to derive different mathematical notations to anticipate the truth assumption of each.

After computing the probabilistic weight of all model parameters, we used the predefined values to enhance the graphical representation of the developed model. The obtained values were translated into shades of gray to improve the visual expressiveness of BBFM.

To demonstrate our approach, we applied our analysis on different models, and successfully validate the obtained results.

BBFM is a pioneering approach for modeling under uncertainty in SPL, in which;

1. Each parameter is assigned a probabilistic weight, quantifying its actual implication on the model belief.
2. The truth assumption of any feature is a result of the feature specialization and generalization dependency flow. Therefore, by evaluating the truth assumption of each feature, we will be able to trace the truth flow of this feature throughout the model. In other words, we will be able to measure the feature indirect interaction with other features, and estimate any nonfunctional interaction might arise.
3. Due to the uncertainty measure, incorporating and removing new features would be more scalable and traceable in BBFM; as long as we maintain the existing semantics.

The use of colour, made BBFM more Self-expressive, in a fashion were we can understand the dependency flow, and the truth value of models' parameters by simply tracing the colour variation throughout the model

Part V

Reasoning

Chapter 6

Reasoning under Uncertainty

Reasoning about alternative decisions is a perplexity which presents a problematic challenge to the Computer Science and Artificial intelligence community. Ever since the earliest endeavors, there has always been a tendency between scientists and researchers concerning the complications in alternatives and variable choices, which are cross related to the complications in reasoning. The high complexity in variables and dependencies amongst them points out conventional presentations and reasoning methods are profoundly amiss. This chapter intends to bring about a contribution to the state of the art in regards to this challenge.

6.1. Introduction

In Bayesian Belief Feature Models (BBFM), different variables represent different functionality in the knowledge domain. Variables are grouped into different dependency contexts; in such a way each context implies different logical interaction.

BBFM may need to reason about its own knowledge in the problem space, while maintaining the proposed semantics in the introduced model.

Reasoning in Bayesian Belief Feature Model refers to the idea that the model take into account not only core features in the problem space, but also other non-core features identifying different legal configurations with different functionalities.

Whilst reasoning, decisions are made to choose between different variables and search for alternatives that are consistent with introduced dependency contexts and any involved constraints.

Due to the dependency nature and problem's complexity, variables need to be grouped in different settings. When sets of variables interact with each other in a logically predefined dependency, whilst sharing the same ancestor variables, these set of variables form a dependency context.

In account, dependency context is a logical representation for a set of variables, in which the truth assignment of these variables is derived using propositional logic axioms and inference rules.

By reason of the problem complexity and high level of dependency among variables, classical logic gates fails to capture further interaction between variables from different dependency contexts. All correlations among variables are imperative and thus must be treated crucially, which infers it should be satisfied when evaluating the truth assignment of involved variables. To avoid any contradiction in the belief model, cross-tree constraints are introduced to the problem space model, Such that it captures further dependencies among variables from different dependency contexts, or enriches the existence semantic in a given dependency context to avoid any information loss that subsequently might advance a false truth conclusion. Truth conclusion is identified truth assignment for a set of variables in an explicit configuration, such that the resulting evaluation is valid, complete and consistent with the model predefined dependencies.

Not all truth assumptions are satisfiable even under the supposition it was initially consistent with the introduced dependency context.

In consequence, when choosing a feature with given truth assumption, it is vital to understand the implication of this assignment on the model behavior, taking into consideration whether this assumption is supporting the constraints satisfaction problem.

Moreover, decisions are highly influenced by the system requirements and the model structure.

Accordingly, system requirements are a set of system qualities that are desirable by the stakeholder or the design engineer; some of these qualities are primarily a matter of possible functionalities, cost, efficiency, degree of variability, system complexity, model scalability, configuration stability...etc.

Decisions are made to assign a truth value for any set of variables, these decisions contains a set of preferred features that are ought to be present in the product final configuration. However, different decisions aggregate to expose alternatives that are most apt to the dependency flow and some preferred qualities and functionalities.

Uncertain occurrence of model variables coupled with the problems' degree of variability in BBFM increase the complexity of the constraints satisfaction problem in SPLE.

This chapter addresses this problem as *Uncertain Constraints Satisfaction Problem.* Unlike traditional constraints satisfaction problems, uncertain constraint satisfaction problem reason about the problem space with a degree of belief, and rectify the imposed belief by exposing it to the predefined constraints. Due to the uncertainty nature of the satisfiability problem, probabilistic weight is imposed for each parameter throughout defining a range of satisfiable assignments that are consistent with the involved dependency context. Intrinsically, different probabilistic weights mean different degree of belief. Also, higher degree of belief means higher certainty level. This formulates a quite interesting observation, which is in fact actually perceived everyday with real life problem. When encountering any problem carrying out uncertainty on how to sort, this problem is firstly identified and analyzed followed by an attempt to investigate all involved contributing factors. Accordingly, this builds up having more knowledge about the problem, and the more knowledge we have the higher our belief goes and the closer we come to the solution.

In the proposed approach, the problem is handled in the same concept. First, we understand the problem dependency context as been modeled via BBFM. Then, we compute the probabilistic weight of model parameters, to anticipate its' occurrence chances. Afterwards, we quantify the implication of the feature on empowering constraints. Meaning, the higher the occurrence probability of any features the stronger its constraints implication will be. In other words, if we find out that a specific feature has a high occurrence probabilistic weight, we can conclude that the implication of its involved constraints on the models' parameters will be higher. Respectively and in response to the aforementioned, any imposed belief would be continuously revised in order to anticipate any proposed assignments' success possibility.

The proposed approach is centered on three different orientations; observation, decision and conclusion.

The aforesaid are outlined as follows:

- **Observation:** Reflection is given on truth assignment for a set features, in which we assign each feature maximum truth-value (in which probability weight equal one) to include it in all possible configurations.
- **Decision:** Decision tends to be a domain of interest, in which we question the truth assumption of a given parameter to anticipate its existence on the final products configurations.
- **Conclusion:** Conclusion settles which decisions are satisifiable and consistent with the model belief.

6.1.1. Further Highlights

This chapter tackles the problem of constraint satisfactions in SPLE by considering the uncertainty of model parameters. Through predefined probability distribution that has been obtained in using the introduced mathematical notations. The probability distribution forms the benchmark of the model belief, confirming the intended semantic of all dependency contexts and the dependency flow thorough out the Bayesian Belief Feature Model. When reasoning, we assign new truth-values of the features that are to be included perpetually on the product configurations.

Consequently, this will pilot to a new truth flow in the belief model; therefore, new probabilistic distribution of all affected parameters. The new truth assignment is called '*hypothesis*' in which we enclose a set of features that are required by the system-preferred configuration, whiles aiding in the reasoning efficiency. Moreover, when including any feature in the hypothesis set, it is necessary to understand how this selection utilities the problem satisfiability behavior. Therefore, it's preferred to include variables that imply a truth flow to some of the constraint, to help exposing the constraints implication on the model behavior, or to simply intensify or soothe the constraints implications on other parameters.

In addition, this chapter addresses the uncertain constraints satisfaction problem as an automated reasoner of the previously introduced Bayesian belief feature model.

- Firstly, we provide an overview to the uncertain satisfaction problem alongside its properties and assets.
- This will be tracked by our approach of reasoning under uncertainty. Our proposed approach will consider factorizing all constraints and extract sub-problems.
- Consequently, Assumptions that facilitate reducing the problem search size will be presented, after extracting the sub-models.
- This is preceded by launching our optimization method to improve the reasoning process in the next subsection.
- Successively, our approaches are tested by conducting in depth reflective experiments to testify the effectiveness of our methods, as well as analyze the results.
- We will finally provide a short summary in which we recapitulate and wrap up this chapter by presenting headline of the used approaches and enlighten on the aspects of the analysis and findings.

6.2. Uncertain Constraints Satisfaction Problem

Constraint satisfaction problem comprises of set of variables, each with valid truth domain, along with a set of constraints that confine the encapsulated truth domains

when called. Each constraint must have valid truth domains that are coherent with the involved variables' truth assumptions. Constraints can be seen as a set of valid assignment of a given variables in accordance with the truth assumption of another correlated variable. In an uncertain constraints satisfaction problem, we drive the implication of given constraints each by exposing the possible outcomes of its dependency context. In addition, instead of constraining whole dependency context, constraints will take effect only on affected domains, while unaffected domains would be assigned a truth assumption in accordance to its dependency context.
Accordingly, a series of definitions associated with the uncertain constraints satisfaction are expounded as below:

Definition 6.0. An uncertain constraints satisfaction problem is *5-uple* $P = \langle H, \lambda, V, C, \omega \rangle$ *where:*

- $H = \langle x_1, x_2 .. x_n \rangle$ Is a set of belief hypothesis (observation set), such that x will be assigned a certain truth assignment equal 1.
- $\lambda = \langle \lambda_1, \lambda_2 .. \lambda_n \rangle$ Is a set of dependency contexts with satisfiable domains (regardless cross-tree constraint). λ is descendent of x and might arise a mutual cross tree dependency with another dependency context λ that is also a descendent of the same common ancestor variable x.
- $V = \langle v_1, v_2 .. v_n \rangle$ Is a set of interconnected variables, in which its truth assumption and satisfiability domain is defined by the semantic of its dependency context λ.
- $C = \langle c_1, c_2 .. c_n \rangle$ Is a set of mutual dependency constraints (cross-tree constraints), such that each vertex c has two edges forming a mutual interaction between two variables v.
- ω Is a revised a probabilistic weight of v and λ, after infusing the new observation hypothesis H in the problem belief.

When including variable x in our belief hypothesis *H,* in which x forms a common ancestor of at least two variables v_1 and v_2 from two different dependency contexts λ_1 and λ_2 with a mutual cross–tree dependency; constraining the truth domain of both. There's at least one unsatisfiable assignment that might jeopardies our belief assumption and invalidate it, which leaves us with the possibility of obtaining false assignment. Hence, there are some assignments that will not evaluate x to be 1, even though all its fellow dependency contexts λ were initially consistent and satisfied.

When enabling x with new truth assignment, such that $\omega = 1$; a flow of truth assumptions' adjustments would be enforced on all x specializations. This indeed would infer a possible increment on the involved constraints' implications.

In addition, all consequent dependency contexts must be satisfied and consistent with the new belief upgrade. Therefore, all invalid truth domains must be excluded from the problem truth space.

A set of complete, consistent and valid assignments of a group of variables *V* is called world *W*. World *W,* is legal truth assignment of subset of the set of all variables *V*, such that $W \subseteq V$, in which all truth assumption of all variables in the subset *W* must be consistent with the semantic of its dependency context, regardless of any mutual dependency that *V* might exercise.

Definition 6.1. Let *P* be an *Uncertain CSP* , with a set of variables grouped within the same dependency context. Possible World *Poss*(*W*), is a truth space of all valid assignments that satisfies the logical semantics of the involved dependency contexts, after ignoring any constraints $POSS(W) = \langle w_1, w_2, w_n \rangle$ such that $P(w) > 0$.

All possible worlds must be consistent with its dependency context. Also, some of these worlds might exercise mutual dependency with other dependency contexts. This positively might reveal a false consistency in the problem space jeopardizing the satisfiability assignments; possible world with mutual dependencies are to be called Candidate Problem.

Definition 6.2. Let *P* be an U*ncertain CSP* with a set of possible worlds *Poss*(*W*). *Candidate Problem* is a subset of set of all possible worlds

Candidate Problem $\subseteq$ *Possible Worlds*, such that each event entangled at least with one mutual dependency via crosstree constraints.

Each *Candidate Problem* entitled a set of satisfiable assignments called *Candidate Domains.* Thus, they are compatible with the truth semantic of the involved crosstree constraints.

Definition 6.3. Let *P* be an *uncertain CSP* with set of *Candidate Problem*. *Candidate Domains* are set of truth assumptions aroused in candidate problem, such that, each truth assumption is consistent with the embedded semantic of the involved crosstree constraints.

Definition 6.4. We define the satisfiability factor as the ratio of Candidate Domain over Candidate,

$$\text{Problem, } \textit{satisfiability factor} = \frac{\textit{Candidate Domain}}{\textit{Candidate Problem}}.$$

Proposition 6.1. A problem with satisfiability factor equal one is undetermined problem, such that the number of existence consistent assignments are not sufficient enough to satisfy the given crosstree semantic.

Proof and semantic Proof by induction. Suppose we have a consistent sub-problem with crosstree constraints, in which the maximum number of satisfiable domains without considering the crosstree constraints' implications equal *V.* When considering the crosstree constraints implication, the maximum number of satisfiable domains would drop to *V-R,* such that *R* is candidate domains. The difference between number of satisfiable domains before and after introducing the constraint implications is equal the number of constraints involved domains $V-(V-R)=R$, when no implication is found then *R=0* and $V-(V-0)=0$. This semantically holds with our claim as the difference equal zero meaning no change occurred.

The maximum difference $V-(V-R)$ can be gained, when $V=R$.

Suggesting, the maximum number of domains that can be imposed by constraint would equal candidate problem.

In that case, number of satisfiable domains would become zero meaning no satisfiable domains can be found $V - R = V - V = 0$.

Therefore, if $R = V$ similarly $\frac{R}{V} = 1$, no satisfiable domains can be found identifying undeterministic problem with satisfiability factor equal 1, which semantically proves the proposition.

6.3. Problem Extraction and constraints factorization

Features in SPLE are base 2 binary numeral variables, with a truth domain of zero and one for each singular variable. However, the truth assignments for one feature doesn't necessary imply a truth assignment of its dependent feature (specialization or generalization). Nevertheless, it's critical to determining the truth assignment of its mutual dependent feature (cross-tree constrained). That being said, due to the evidential and casual truth flow in BBFM, when observing a truth assignment of any feature, the overall model belief would probably respond to the new observation, concluding a new truth assumption for some dependent features (specializations and generalizations). The new truth assumptions can only be quantified after evaluating the dependency semantics of the involved dependency contexts.

While the truth assignment of most features can be determined by singular observation; the truth assignment of Dependency Contexts is more demanding and only can be assessed through multiple observations. Generally, the truth assignment of Bayesian Conjunction and Bayesian Exclusive disjunction dependency contexts can be evaluated by a set of truth assumptions for all involved features, extending the binary truth assignment into a domain of truth assignment. This is unlike the case of Bayesian Disjunction and Tautology Dependency context, in which its truth assignment can be concluded from one truth observation.

When grouping set of features in a certain dependency context, the anticipated truth assumptions might indicate the context truth domain. The resultant domain is a set of valid assignments that are consistent with the dependency context, coupled with invalid truth assumptions, which are inconsistent with the involved dependency context. Any obtained truth assumption is due to belief base, such as per core features and observation set, and dependencys' flow (features specializations, generalizations and mutual dependencies).

Different degrees of variability develop different truth domains. Due to the dependencies semantics, truth boundaries take place to limit the model behavior and formulate the valid truth sets for any dependency context.

For an instance, any truth assumption of a given feature must be consistent with dependency contexts' valid truth domain as well as any coupled mutual dependency contexts' truth domain.

To satisfy both truth domains, a subset of the set of the aggregated truth domains must be excluded from the model. This requires evaluation of the truth assumption of all involved parameters, making the reasoning process one of the most complex and expensive challenges in SPLE with ultimately an NP-complete satisfiability problem.

Beforehand, weighting techniques to anticipate the truth assumption and the occurrence likelihood from one valid domain to another is by now introduced and acquainted with.

In this chapter, we are going to extend our work by employing the acquired probabilistic value in the reasoning process; emphasizing on reasoning under belief uncertainty. Definition 6.0 specifies the attributes of uncertain constraints satisfaction problem Uncertain CSP. When observing a new feature x in the problems' belief hypothesis H, this observation will advance our belief by assigning a truth assignment of x and set its probabilistic weight to be 1.

To satisfy the new change in the model belief, only valid truth domains must be included in the feature specializations; that are consistent with the new observation. Moreover, any additional mutual dependency must be satisfied and consistent with the new truth domains. This suggests a mutual exclusion for all truth assumptions that are inconsistent with the new belief. Likewise, we guarantee that new observation is robust and satisfiable in all possible configurations.

The new advancement in the model belief will assist in reducing the problem space, through discounting a set of inconsistent truth domains, which in its turn, enhance the reasoning performance.

In general, any truth assumption that's consistent with the involved dependency contexts' truth domain, is in fact consistent with the model belief until it exhibits mutual cross tree dependency, which semantically demand truth exclusion of any assignment that doesn't meet its semantic.

To this interest, we will marginally factories all succeeding dependencies contexts to extract only cross tree constraints involved context. Consequently, we will reason about truth domains with cross tree entanglements.

Definition 6.5. Let *W* be a possible world in *Uncertain Constraints Satisfaction Problem P*. *W* has world weight such that,

$$\textit{world weight} \in (0,1]$$

whereas, world weight represents the probability weight of *W* occurrence.

Definition 6.6. Let *W* be a possible world in *Uncertain Constraints Satisfaction Problem P*. *W* has a satisfiability weight Sat weight such that,

$$\textit{Sat weight} \in [0,1]$$

Whereas, Sat weight is a probabilistic measurements quantifying the truth weight of all possible assignments that contain *W* and are consistent with all cross tree constraints truth domain.

Alternatively, *Sat Weight* is the probability of having a satisfiable and only satisfiable assignment of all truth domains that contain world *W*.

If *Sat Weight* equal zero, this ultimately means *W* is not consistent with the model belief, therefor will not appear in any product configuration.

However, If *Sat Weight* equal one this means *W* is always satisfiable. Concluding that any truth assumption contain *W* in its truth domain, such that $\{W\} \in \{\textit{truth domain}\}$ is ultimately can be satisfied.

- **Semantic Differentiations between definitions 6.5 and 6.6**

It is evidential that the more facts we know about any parameter, the more accurate our anticipation about the parameter's truth assignment will be.

When modeling, we can predict the probability of having a certain parameter in a given set according to its dependency context, and the type of dependency correlation it has with its ancestor parameter; through conditioning its existence by the existence of other pre-specified parameter as have been utilized by Bayes' theorem.

Such a notion can be denoted as $P(f_c \mid f_p)$, whereas f_c is a child feature and f_p is its ancestor parent feature. This notion is of high practicality in estimating the likelihood of selecting one feature by itself or among other sibling features. It is also highly beneficial during the modeling phase to formulate our selection's preference throughout choosing dependency contexts that are more likely to conclude the desired functionality by evaluating its valid truth domains. In addition, it's vital to derive the dependency semantic among set of sibling features.

In Definition 6.5 we outlined this notion as *world weigh*t. Nonetheless, it is indispensable to argue whether this notion sufficed to capture the truth state of the given parameter or not. The response for this argument would inevitably be almost certainly yes, until we perceive new information about some added constraints, or in other case, until we observed that this parameter has a degree of association with some cross tree constraints.

New information means new belief, which in its turn means a forward step toward understanding the actual behavior of the parameter truth assignment. The new information that contains a constraints association is captured by definition 6.6. In which we anticipate the truth assumption that's not only consistent with the involved dependency context but also meets the crosstree dependency semantics.

In other words, we quantify the probability of getting a specific feature in the resulting satisfiable configuration by opting out all its inconsistent assignments.

This notion is unquestionably more accurate than the former one, in which false truth assumptions would be excluded from the prediction set.

Observing a constraint would change the truth performance of any associated feature. In fact, it might conclude that a certain parameter or set of parameters will never obtain a truth assignment on the reasoning process or the exact opposite.

The developed approach is deliberately studied in accordance to a designated example as follows:

Example 6.1. Consider the following *Uncertain Constraint Satisfaction Problem P*. of a given sub-problem of Bayesian Belief Feature Model. Such that,

- $H = \langle x_1 \rangle$
- $\lambda = \langle \lambda_+, \lambda_\oplus \rangle$
- $V = \langle (a,b)_+, (c,d)_\oplus \rangle$
- $C = \langle (b \wedge c) = \varnothing \rangle$
- $\omega = \langle 0.5, 0.5, 0.125, 0.125 \rangle$ For *a,b,c and d* respectively.

λ_+	*Cand(P)*
a	T
b	T
ab	T
nil	F

$\lambda_\oplus$	*Cand(P)*
c	T
d	T
cd	F
nil	T

c		*Cons(P)*
c	*a*	T
c	*b*	F
c	*ab*	F

b		*Cons(P)*
b	*c*	F
b	*d*	T
b	*nil*	T

ab		*Cons(P)*
ab	*c*	F
ab	*d*	T
ab	*nil*	T

Table 6. 1 Dependency Contexts and Constraints Truth Domain

In example 6.1, the validity of any selected world of interest is checked to testify its consistency with the model belief among the existence crosstree constraint.

When choosing any world *w* from the problem truth space, we start by breaking this world into subsets *x* suchthat *w* is an element of *x* $w \in x$. After defining all possible subsets; a consistency check is conducted, in which any subset that is not consistent with the provided dependency context got excluded.

Now we establish a set of subsets that they are all;

i. Consistent with their dependency contexts.
ii. Entangled with world *w*.

We now advance the consistency check of the obtained set by factorizing each subset with its corresponding constraint, to exclude any subset that doesn't satisfy the constraint truth domain, and the problem semantic. The attained set formulates the problem satisfiable domain, likewise we have determined all possible and satisfiable truth domains that are consistent with the problem dependency semantic, and don't break the infused constraints semantic. Using the predefined probabilistic weight of all resulting subsets we can simply compute the satweight of world *w*.

The aforementioned technique describes the general functionality workflow of the developed algorithm. The algorithm is sketched as Algorithm 1. 1.

Lets consider world *w* , such that $W_1 = \langle a,c \rangle$; by running Algorithm 1. 1 we can breakdown the reasoning process as in the following:

- Decompose world *w* into subset, we get $\langle a,ab;c,cd \rangle$.
- Exclude all inconsistent subsets, which fails to meet the dependency contexts' semantic, we get $\langle a,ab;c \rangle$.
- Next, aggregate the dependency contexts by product all consistent subset and then remove any duplication, to get $\langle ac,abc \rangle$
- Thenceforth, check the compatibility of the obtained set by factorizing it with the constraints truth space, in which one subset doesn't imply a valid truth assumption and is excluded from the set, thus the remaining subsets formulate the world *w* sastsifiable domain. Such that, satdomain is $\langle ac \rangle$

- Given the predefined probability distribution, we now can compute the satweight such that; satweight equal $\langle \Pr(a) \times \Pr(c) x \Pr(ca) \rangle = \langle 0.5 \times 0.125 \times 0.333 \rangle = 0.0208$
- Finally, the algorithm return $\langle \langle ac \rangle, 0.0208 \rangle$

This allow us to conclude that if we want to reason about $\langle a,c \rangle$, there's only one domain out of all possible domains of $\langle a,c \rangle$ that is satisfiable and consistent with all constraints and local dependency with a probabilistic weight of obtaining this world satweight, suchthat satweight equal 0.0208 .

Now when considering a new world, such that $W_2 = \langle b \rangle$, the algorithm will return $\langle \langle b, ab \rangle, 0.1666 \rangle$.

On another hand, when reason about a new world such that $W_3 = \langle b,c \rangle$, the algorithm will not return any satisfiable domain, implying that this world is not compatible with the problem's overall semantic, therefore doesn't qualify any valid assignment.

Clearly W_3 is a bad decision and should be avoided when choosing world of interest.

```
Choose w
$\langle Dec \rangle := Decompose \langle w \rangle$
$\langle Cons \rangle := Consistent \langle Dec \rangle$
    i =1 ; j =1
    Repeat
        $\forall E \in \langle Cons \rangle$
        $P_j := E_i * (\langle Cons \rangle - (E \subset E_i))$
        $i = i + 1$
        $P_j \rightarrow Product \langle P \rangle$
        $j = j + 1$
    until$(i = n)$;
$Product \langle P \rangle :=$ Delete duplication $Product \langle P \rangle$
$SatDomain \langle P \rangle := Factorization \langle \forall P \in Product \langle P \rangle \bigcup Constraints \rangle$
    $z = 1$ ;$Satweight = 0$
    Repeat
        $\forall P \in SatDomain \langle P \rangle$
        $Satweight = Satweight + productweight(P_z)$
        $z = z + 1$
    until$(z = j)$;
$Return(SatDomain \langle P \rangle, Satweight)$
```

Algorithm 1. 1 *Uncertain CSP* reasoning algorithm

World W	*World weight*	*Satisfiable Domain*	*Sat weight*
$\langle a\rangle$	0.5	$\{a,ab\}$	0.333
$\langle b\rangle$	0.5	$\{b,ab\}$	0.166
$\langle ab\rangle$	0.25	$\{ab\}$	0.0833
$\langle a,c\rangle$	0.0625	$\{ac\}$	0.0208
$\langle b,c\rangle$	0.0625	*nil*	0
$\langle ab,c\rangle$	0.03125	*nil*	0
$\langle a,d\rangle$	0.0625	$\{ad,abd\}$	0.0416
$\langle b,d\rangle$	0.0625	$\{bd,abd\}$	0.0208
$\langle ab,d\rangle$	0.03125	$\{abd\}$	0.0104

Table 6. 2 the obtained sat weight value for different Worlds

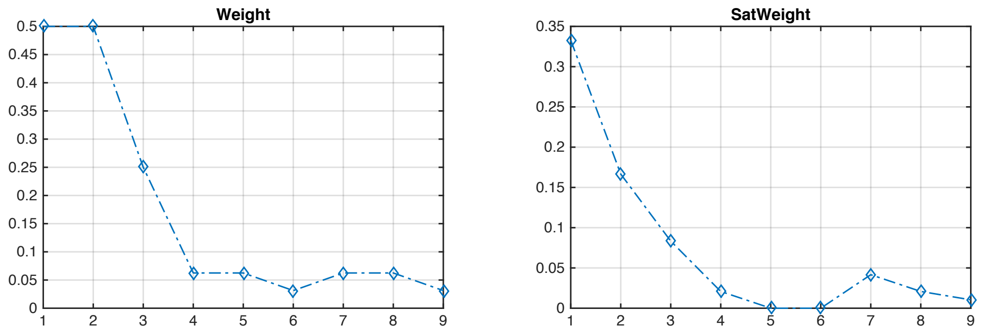


Figure 6. 1 measure difference between weight and satweight

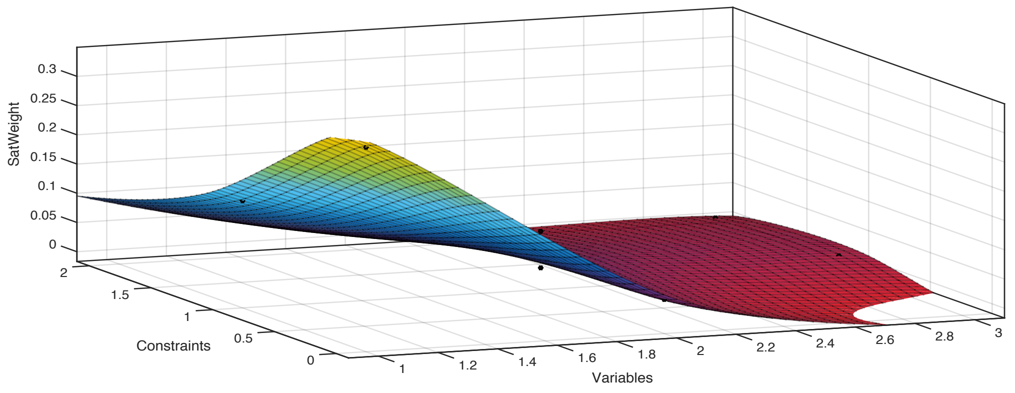


Figure 6. 2 Satweight response versus change in number of variables and number of constraints

Lemma 6.1. Let P be *Uncertain Constraints Satisfaction Problem P* with no cross tree constraints involved such that $C := \{nil\}$. P is satisfiable among all possible worlds.

To study lemma 6.1, we are going to revise the above analysis of example 6.1 while ignoring the implication of the involved crosstree constraints, if Lemma 6.1 holds this mean all possible worlds must obtain a valid sat weight value.

In according to the preceding approach, we achieve the following outcomes, as per Table below which proves the validity of Lemma 6.1.

World W	*World weight*	*Satisfiable Domain*	*Sat weight*
$\langle a \rangle$	0.5	$\{a,ab\}$	0.5
$\langle b \rangle$	0.5	$\{b,ab\}$	0.5
$\langle ab \rangle$	0.25	$\{ab\}$	0.25
$\langle a,c \rangle$	0.0625	$\{ac,abc\}$	0.0625
$\langle b,c \rangle$	0.0625	$\{bc,abc\}$	0.0625
$\langle ab,c \rangle$	0.03125	$\{abc\}$	0.03125
$\langle a,d \rangle$	0.0625	$\{ad,abd\}$	0.0625
$\langle b,d \rangle$	0.0625	$\{bd,abd\}$	0.0625
$\langle ab,d \rangle$	0.03125	$\{abd\}$	0.03125

Table the obtained satweight values when ignoring the constraints implications

Definition 6.7. Let W be a world of interest such that $\langle x \rangle \in W$. The ratio between the satisfiability weight and world weight of x is defined as the success ratio of x.

$$success\ ratio(x) = \frac{sat(x)}{weight(x)}$$

In Table 6. 3, we present the obtained success ratio of example 6.1.

World W	*World weight*	*Satisfiable Domain*	*Sat weight*	*Success Ratio*
$\langle a \rangle$	0.5	$\{a,ab\}$	0.333	0.666
$\langle b \rangle$	0.5	$\{b,ab\}$	0.166	0.333
$\langle ab \rangle$	0.25	$\{ab\}$	0.0833	0.333
$\langle a,c \rangle$	0.0625	$\{ac\}$	0.0208	0.333
$\langle b,c \rangle$	0.0625	nil	0	0
$\langle ab,c \rangle$	0.03125	nil	0	0
$\langle a,d \rangle$	0.0625	$\{ad,abd\}$	0.0416	0.666
$\langle b,d \rangle$	0.0625	$\{bd,abd\}$	0.0208	0.333
$\langle ab,d \rangle$	0.03125	$\{abd\}$	0.0104	0.333

Table 6. 3 obtained success ratio of example 6.1

The analysis commenced delivers a number of interesting properties explained as below:

- By definition, we know that worlds with less degree of association with crosstree constraint exhibit higher success ratio.
- On one hand, if the worlds' set of all subsets doesn't exhibit any crosstree mutual dependency, such that $\forall W(x), x \textit{ Has no constraints}$, this would evaluate success ratio equal 1. Implying that, any subset of all subset $x \in W(x)$ is valid satisfiable, which also holds by the semantic of Lemma 6.1.
- On a different scenario, whereas any subset of the set of all subset is associated with crosstree constraint, the success ratio will drop accordingly, which is a result of the semantic of the involved context association.
- It is also noticeable that success ratio drops in anticipated patterns.

In example 6.1, we can conclude that if the world *W* isn't directly entangled with crosstree dependency constraint in a way that this world is a vertex of the constraint edge, then the success ratio will be as twice as the success ratio of the world that has a direct entanglement with crosstree dependency constraint.

Moreover, different worlds with same degree of association with crosstree constrains produce the same success ratio.

Finally, if all subsets of *World W* were entangled with a crosstree constraint, *W* is inconsistent with the crosstree dependency semantic such that Satweight and success ratio would be zero,which also holds by the semantic of propostion 6.1.

More analysis would be observed and explicated in the experimental section.

It is important to shed the light on this analysis as it is very vital to optimize the satisfiabilty problem, which is also minimal comparing with the actual problem size hence we reason about domain of interests.

When deciding what world would is fit to be included in belief hypothesis, success ratio of this world might be a key quality among other qualities.

Definition 6.8. We define dominant feature as feature or set of features *x*,such that:

i. x is subset of the set of the possible worlds $x \in \{Poss(W)\}$.
ii. The success ratio of world x is equal one.

Proposition 6.2. If Possible world *W*, contained a dominant feature *x*, such that $x \in w$ then w is satisfiable with satisfiability weight greater than zero.

Definition 6.9. A decision *s* is a subjective choice of selecting a possible world to reason about. Decision *s* is bad decision such that the success ratio of $W(s)=0$.

In example 6.1, both of $\langle b,c \rangle$ and $\langle ab,c \rangle$ are bad decisions and the only way to optimize this decision is by changing constraints dependency semantic. This might not be an option in most cases, especially at advanced stages as it might change the product semantic dramatically and also expensive.
Henceforth, it is a recommended to avoid bad decisions completely in the reasoning process.

Definition 6.10. A decision s is good decision, only and if only, the success ratio of $W(s) > 0$. Decision s could be optimal decision if its success ratio was the highest obtained ratio in the success ratios table.

In example 6.1, $\langle a \rangle$ is an ultimate decision. Decision s could be a dominant decision if its success ratio was equal one, which only occurred if the satisfiable domain doesn't have any degree of association with the introduced crosstree constraints.

6.4. Optimizing Under Uncertainty

In this section, selected methods to optimize the decision are introduced in order to find good decisions s efficiently, and obtain higher success ratio. As evident, finding an ultimate decision could be computationally costly; therefore the main focus in this section is to attempt improving the success ratio values of all worlds to achieve this goal with minimal cost.

When it comes to the semantic of dependency context, it is mostly flexible, in which different contexts have different degree of variability.
Deploying some assumption in the original semantic, without breaking the context consistency, might be useful to exclude some invalid domains from the reasoning evaluation and decrease the problem space. The most flexible semantic can be found in the tautology dependency context. On the contrary, conjunction dependency context is the stiffest dependency context.

Definition 6.11. Degree of variability DoV, is a quality of dependency context that indicates the number possibly valid assignments for any set of variables grouped in this dependency context, such that these assignments are legal and consistent with dependency context semantic.

If we have a set of features x such that $x = \{x_1, x_2 .. x_n\}$, we might group these features within different dependency contexts to obtain deferent satisfiable truth domains. The key matter in this regard is the common domain. That being said, we can find common domains when producing different dependency contexts. This quality is to be beneficial, if we want reduce the search space for the optimal decision while maintaining the original semantic of the dependency context and narrowing the search space of the produced truth domain.

A profound investigation on some possible assumption and testing its reliability is undertaken as follows:

Assumption 6.1. In a set of all subsets of features, in which these features are connected via Bayesian Disjunction Dependency, we might minimize the problem space by reasoning about truth domains that arise via Bayesian exclusive disjunction dependency context. i.e considering that all features are mutually independent of each other's.
We argue that this assumption will optimize the reasoning process by reducing the search space for good decision and ultimate decisions, without violating the semantic requirements of the original problem context. However, employing assumption 1 will conclude a naïve truth assumption for the actual problem truth space.

In example 6.1, we have two dependency contexts. λ_{+} and $\lambda_{\oplus}$. When employing assumption 6.1 both context would be Bayesian exclusive disjunction contexts , such that;

$\lambda_{\oplus}$	*Cand(P)*
a	T
b	T
ab	F
nil	F

$\lambda_{\oplus}$	*Cand(P)*
c	T
d	T
cd	F
nil	T

Table 6. 4 dependency contexts truth domains

Although the mutual dependency semantic is still the same, constraints factorization will be different. Such that;

c		*Cons(P)*
c	*a*	T
c	*b*	F

b		*Cons(P)*
b	*c*	F
b	*d*	T
b	*nil*	T

Table 6. 5 constraints truth domain

Table 6. 6 demonstrates the results achieved as per obeying the same reasoning and pruning process:

World W	*World weight*	*Satisfiable Domain*	*Sat weight*	*Success Ratio*
$\langle a \rangle$	0.333	$\{a\}$	0.333	1
$\langle b \rangle$	0.333	$\{b\}$	0.111	0.333
$\langle a,c \rangle$	0.041625	$\{ac\}$	0.0208	0.5
$\langle b,c \rangle$	0.041625	*nil*	0	0
$\langle a,d \rangle$	0.041625	$\{ad\}$	0.041625	1
$\langle b,d \rangle$	0.041625	$\{bd\}$	0.013875	0.333

Table 6. 6 obtained results after the reasoning about all possible worlds

As a result of employing assumption 6.1 in the given example, the implication of crosstree constraints has been subsided, concluding an increment in the problem success ratios as shown in Table 6. 6.

In the aforementioned example, the obtained decisions were two ultimate decision, five good decisions and two bad decisions. However, when applying assumption 6.1, the obtained decisions were two dominate decisions, three good decisions and one bad decision.

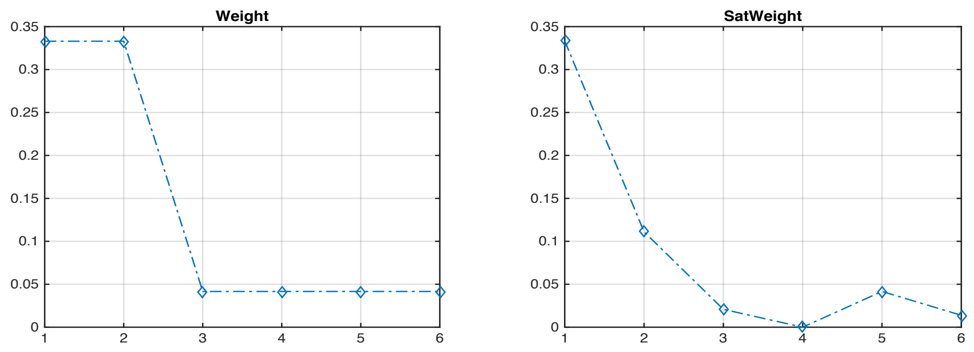


Figure 6. 3 measure difference between weight and satweight

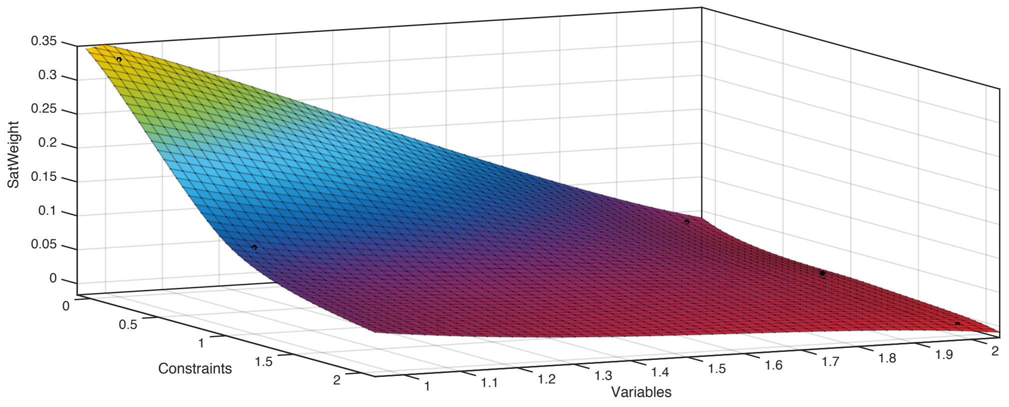


Figure 6. 4 satweight response versus change in number of variables and number of constraints

In the original problem space seven out of nine (almost 77%) decisions produced satisfied domain. Whereas, after deploying assumption 6.1, five out of six (almost 83%) decisions produced satisfied domains. Hence, the probability of obtaining a satisfiable domain is to be increased when employing assumption1.

In addition to that, after implementing assumption 6.1; the problem search space has dropped by almost 33% after. This constructively fosters towards reducing the computation cost significantly, particularly in large scale problems.

Moreover, all new obtained satisfiable domains are consistent and valid in the original dependency context. As per our example, 71% of the originally satisfiable domains can still be found even after applying assumption 6.1.

However, even though substantial advantages are present, there's a single downfall related to the assumption. This distinct downfall of recalling assumption 6.1 is the reduction of problem degree of variability, and consequently the reduction of problem number valid domains when compared with the original dependency context.

6.5. Searching for optimal decision

Undoubtedly, rummaging for optimal decision is known to be computationally costly, as the problem space in worst scenarios could be exponential. Ultimately, some problem might be NP complete problem. For instance, suppose we have 20 features divided equally into two sets, such that each set grouped within Bayesian disjunction dependency contexts. Moreover both contexts are also connected in Bayesian disjunction dependency context. Given the fact that, there are some features in both sets exhibit a mutual crosstree dependency.

Although we started with only 20 features with binary truth assignment for each, we will conclude more than one million possible worlds due to the degree of variability of the valid truth domains. Apparently, reasoning among such massive number can be costly.
Problems with large search space motivate us to propose a new method in line of finding conditional decisions in more practical way.

Definition 6.12. When employing assumption 6.1 in *Uncertain CSP P*, the world with highest obtained assignment is called conditional decision.

In large size problems, assumption 6.1 is to be employed, as it allows a significant decrease in the number of possible worlds. For an instance, in the aforementioned example, we might be able decrease the problem size into almost only 1000 possible worlds. Afterwards, we run Algorithm 1. 1 to find the problem optimal decision. The newly obtained optimal decision will be the conditional decision of the original problem context.

After determining the conditional decision of problem *P*; we might withhold assumption 6.1 from the problem definition, consequently return to the problem original definition, while running our algorithm subjectively. When choosing a new world *w*, we might or might not consider worlds that are correlated with conditional decisions potential.
In this condition, The key difference that will play a new role is that we already established a new understanding about the anticipated new optimal weight. The new threshold forms the comparison benchmark when quantifying satweight for any new world. We can simply relate to the new threshold and possibly conclude how close we are from the optimal decision. An optimal solution must be equal or less than conditional solution,

$$Satweight(optimal) \leq Satweight(conditional).$$

Corollary 6.1. For a given a dependency context the satisfiability weight for any optimal decision cannot exceed the satisfiability weight of the conditional decision of the same problem. Such that;

$$Satweight(optimal) \leq Satweight(conditional).$$

Building on corollary 6.1, we can now anticipate the behavior of the dependency context more effectively, by setting a new threshold to guide our expectations.

Therefore, when reasoning about any possible world and after finding out the satisfiability weight of this world, we can conclude whether this world produces an optimal decision or not. Moreover, we can also quantify how far this world is from the optimal worlds.

This method can be cost effective, in a sense the search can be stopped whenever we get an optimal world or sense the proximity of an optimal *closed-enough* world.

In this way, time and effort are saved and instead of running through all possible worlds, which might be considerably high number, there's an opportunity to run the algorithm within any time partition and terminate it at any desired time while sustaining the outcome as being informative enough in terms of being optimal or not.

That being said, the satisfiability process can take any time. In addition, experiment results shown that usually the optimal decision satisfiability weight is very close to (90%-100%) to the conditional decision satisfiability weight.

Moreover, it is possible to find some of the unsatisfiable worlds in more cost effectively way. Ultimately, When we recall this technique, we clearly reduce the problem search space, hence we can find some of the unsatifiable worlds in shorter time.

Last but not least, it's observed that worlds that directly get affected by the assumption have either increased its success ratio or at least maintain it.

Example 6.2 Let P be uncertain CSP of a given sub problem of probabilistic feature model. Such that,

$$H = \langle x_1 \rangle$$

$$V = \langle (a,b,c)_+ ,(e,d,f)_+ \rangle$$

$$\lambda = \langle \lambda_+ (\lambda_+ ,\lambda_+) \rangle$$

$$\omega = \left\langle \begin{vmatrix} \lambda_+ & a & b & c \\ \omega & 0.25 & 0.25 & 0.25 \end{vmatrix} , \begin{vmatrix} \lambda_+ & e & d & f \\ \omega & 0.25 & 0.25 & 0.25 \end{vmatrix} \right\rangle$$

$$C = \langle (c) \rangle$$

$$C = \langle (b \wedge c) = \varnothing \rangle$$

This particular example has the highest possible degree of variation for the given set of features. which leaves us with a search space among 63 possible assignments.

When using our approach on all possible assignments, we can obtain the following information:

- Number of unsatifiable assignments = 16 different assignment.
- Assignment with optimal decisions for singular selection is $|\langle a \rangle, \langle b \rangle, \langle ab \rangle, \langle d \rangle, \langle f \rangle, \langle df \rangle|$ with success ratio equal to 0.5625
- Assignment with optimal decisions for multiple selections is $|\langle a,d \rangle, \langle a,f \rangle, \langle b,d \rangle, \langle b,f \rangle|$ with success ratio 0.3125.

Now, in order to reduce our search space, we recall assumption 6.1 to reform our dependency context to become as follow:

$$\lambda = \langle \lambda_+ (\lambda_+ ,\lambda_\oplus) \rangle$$

By recalling this assumption, our search space is narrowed down from 63 to 31 assignments.

When conducting the developed method, the following information will be obtained:

- Number of unsatifiable assignments = 4 different assignment.
- Assignment with optimal decisions for singular selection is $\left|\langle a\rangle,\langle b\rangle,\langle ab,d\rangle,\langle ab,f\rangle\right|$ with success ratio equal to 0.625 and $\langle d\rangle,\langle f\rangle$ with success ratio Equal 1,
- Assignment with optimal decisions for multiple selections is $\left|\langle a,d\rangle,\langle a,f\rangle,\langle b,d\rangle,\langle b,f\rangle\right|$ with success ratio 0.375.

Clearly we can see the resemblance amongst, in much smaller search space (50% smaller) we were able to hunt 8 out of 10 original optimal decision set (80% of the original optimal decisions were obtained). Only two new optimal decisions were generated. However these two new optimal decisions tend to have high degree of optimality even in the original dependency context. This provides a good approximation about the accuracy of our algorithm as well.

Considering this concept, optimal decisions can be achieved more efficiently by reverse running the search process. In this case, instead of looking for the optimal decision directly in the original dependency context and probing through all possible assignments (which might be extremely vast and tedious task), we can employ assumption 6.1 as an initial step and find all optimal decisions set, then point back to the original dependency context and examine the resultant set there.

In the following example we demonstrate our suggested procedure in larger set of features;

Example 6.3. Let *P* be uncertain CSP of a given sub problem of probabilistic feature model. Such that,

$$H = \langle x_1 \rangle$$
$$V = \langle (a,b,c,d,e)_+ ,(f,j,h)_+ \rangle$$
$$\lambda = \langle \lambda_+ (\lambda_+ ,\lambda_+) \rangle$$
$$C = \langle (e \wedge f) \rightarrow \varnothing \rangle$$

To find out the optimal decision of this problem, we need to examine 255 possible assignments, which could be problematic and costly. Using the developed method, the search space ought to be reduced to only 47 (only 18% of the original search space), by considering simply recalling assumption 6.1, on the first set to be $V = \left\langle (a,b,c,d,e)_{\oplus}, (f,j,h)_{+} \right\rangle$.

Afterward, all possible worlds are listed to find out which worlds tend to have optimal value as follows:

- Number of unsatifiable assignments = 4 out of 47 assignments
- Assignments with optimal decisions are $\left| \langle a \rangle, \langle b \rangle, \langle c \rangle, \langle d \rangle \right|$ with success ratio equal 1. And $\left| \langle j \rangle, \langle h \rangle \right|$ with success ratio equal 0.79.
- $\left| \langle a,j \rangle, \langle a,h \rangle, \langle a,hj \rangle, \langle b,j \rangle, \langle b,h \rangle, \langle b,hj \rangle, \langle c,j \rangle, \langle c,h \rangle, \langle c,hj \rangle, \langle d,j \rangle, \langle d,h \rangle, \langle d,hj \rangle \right|$ with success ratio 0.58.

Consequently, the resultant set of assignments is examined in the original dependency context, the results are as follow:

- First group obtained satweight 0.140625 with success ratio 0.5625.
- Second group obtained sat weight 0.12890625 with success ratio 0.515625.
- Third group obtained satweight 0.00933203125 and 0.018066406 with success ratio 0.2890625.

Potentially, when addressing all possible assignments, it is settled that no assignment will have Satweight or success ratio higher than these obtained values, thus making these assignments the optimal assignments.

In the above discussion, we demonstrated a practical technique to optimize the reasoning process and reduce the search space for the developed algorithm.

Since our belief model is based on a set of uncertainties, any new information, measurement, threshold or observation is a forward step toward more accurate anticipation of the actual results.

6.6. Experimental Results

In this section, we present the evaluation *Uncertain CSP* notion. Our method is introduced in which a series of variations of a given *Uncertain CSP P* are experimented to deliver better understanding to the reasoning process, in order to instigate the path of the problem evaluation.

The experiment conducted facilitates the following:

- Pinpoint and define the uncertainty problem *P*.
- Evaluate and experiment the given problem within varying contexts semantics.
- Document and analyze the outcomes.
- Gauge the reliability of our suggested approach and determine its potentials.

By the mean of experimentation, we showcase how the method used is a constructive step towards overcoming the problem addressed.
Furthermore, this section validates how our method can be used derive an efficient reasoning strategies by taking into consideration the desired outputs and reason about it. Understanding the problem behavior, during reasoning process, allow us to modify the problem model in order to respond to the problem functioning and meet the projected results.

At first it is necessary to define and establish the problem to be considered in this experiment;

- **Problem Definition**

Consider the following uncertain constraints problem $P = \langle H, \lambda, V, C, \omega \rangle$ such that;

$$H = \langle X \rangle$$
$$V = \langle (a,b,c),(d,e,f) \rangle$$
$$C = \langle c \wedge e \rightarrow \varnothing \rangle$$

When expending our designated method, we will be able to evaluate the given problem by considering different variations of the same problem by changing the degree of variability 'DoV' within it.

Bearing in mind the degree of variability of any problem is a result of the semantic of the involved dependency context for the given set of variables, it is pragmatic to adjust P by imposing different combinations of different dependency contexts λ, and validate that all possible degree of variability are considered.

In the given experiment, 12 different combinations[4] are to cover all possible degree of variability and any possible combinations of dependency context. Table 6. 7 demonstrates all the experiments' settings of the deliberated sets of dependency contexts.

	Dependency Context			*Dependency Context*	
	$\lambda_{\bullet}$	$\lambda_{\oplus}$		$\lambda_{\bullet}$	$\lambda_{\oplus}$
	$\lambda_{\bullet}$	λ_{+}		$\lambda_{\bullet}$	λ_{+}
$\lambda_{\bullet}$	$\lambda_{\bullet}$	$\lambda_{\bullet}$	λ_{+}	$\lambda_{\bullet}$	$\lambda_{\bullet}$
	λ_{+}	λ_{+}		λ_{+}	λ_{+}
	λ_{+}	$\lambda_{\oplus}$		λ_{+}	$\lambda_{\oplus}$
	$\lambda_{\oplus}$	$\lambda_{\oplus}$		$\lambda_{\oplus}$	$\lambda_{\oplus}$

Table 6. 7 dependency contexts combinations, used in the experiment

It's firmly comprehended that different combination of dependency contexts does not exclusively imply different degree of variability.

Conversely, it also implies different levels of complexity along with different probabilistic weight of the involved variables, in addition to varied implication of the involved crosstree constraints. By varying the dependency context, it is likely to get different numbers of possible truth assignments of the involved variables. Subsequently, number of times that crosstree constraints implication took effect.

[4] This excludes Bayesian Tautology Dependency Contexts, and any sub model using a combination of only Bayesian Exclusive Disjunction Dependency Context.

In this manner, it is feasible to argue that by running the developed method on all possible combinations within varying dependency context, we prove the reliability of this approach and test its capability. In addition, we establish its likelihood to work on any possible kind of problem to be probably encountered in real life application.

Entry 6.1. Firstly, consider the combination with lowest possible degree of variability where only Bayesian Conjunction Dependency Context is allowed in the problem definition such that;

$$\lambda = \left\langle \lambda_{\bullet}(\lambda_{\bullet}, \lambda_{\bullet}) \right\rangle$$

Using our approach, it's noticeable that this problem is unsatisfiable under the given constraints assumption. This result doesn't contradict with the semantic of this problem; in fact, it's rather anticipated and compatible with proposition 6.1, in which we conclude that *P* is undeterministic problem with satisfiability factor equal.

Entry 6.2. We can only have satisfiable assignments by changing the involved dependency context. Thus leading to entry 2 which implements the next variation of *P* such that;

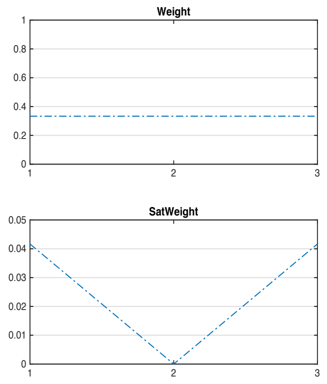
$$\lambda = \left\langle \lambda_{\bullet}(\lambda_{\bullet}, \lambda_{\oplus}) \right\rangle$$

Using our approach, we can conclude that *P* is semantically valid with a possibility of having two different satisfiable assignments, which are legally valid and consistent with the problem semantic.

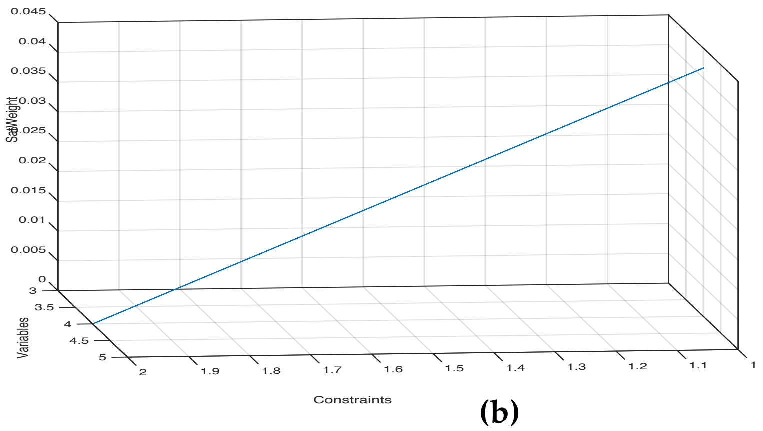
Figure 6. 5 demonstrates the graphical representation of the numerical data as part of the obtained results when using our approach.

By examining Figure 6. 5(a), we will notice that, *P* succeeded two satisfiable assignments and failed to satisfy one assumption.

Figure 6. 5(a) shows the probabilistic weight of all involved assignment *s*. In such a manner, all worlds exhibit a truth assignment when crosstree constraints are ignored, in which the three possible worlds had the same weight, which is equal 0.3333 (refer to Figure 6. 5(a)).



(a)



(b)

Figure 6. 5 a) difference between satweight and weight in entry 6.2
b) Satweight behavior in entry 6.2

When considering the involved constraints, the probabilistic weight of these worlds would drop significantly due to the constraints semantic as shown in SatWeight plot. Accordingly, one world would be detected as unsatisfiable world.

Figure 6. 5(b) illustrates the influence of the number of variable and constraints on the SatWeight value. Based on the plot evidence, we observe that when number of associated constraints is zero, *P* obtains the highest SatWeight assignment. Respectively, the SatWeight value drop when number of associated constraints increases.

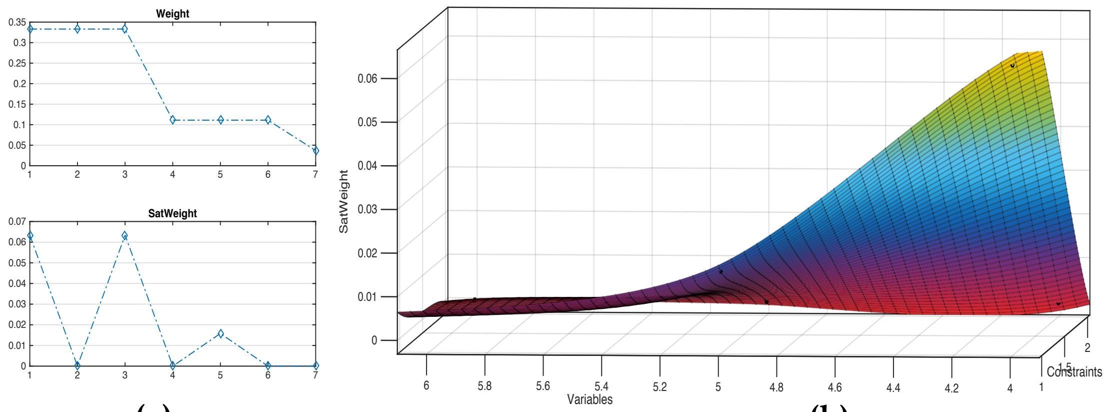
In the given problem settings, numbers of variables were fixed in all worlds. Therefore, it is not feasible to detect the terms in which the change in number of variables affect the SatWeight values.

Accordingly, we manage to understand the behavior of similar problems where the value of SatWeight is always minimal with high potential to have unsatisfiable assignment due to the low degree of variability.

Entry 6.3. Considers the case in which increasing the degree of variability is desired, such that;

$$\lambda = \left\langle \lambda_{\bullet}(\lambda_{\bullet}, \lambda_{+}) \right\rangle$$

Apparently, this consideration will drive more possible assignments due to the increase in the problem degree of variability.



(a) **(b)**

Figure 6. 6 a) difference between satweight and weight in entry 6.3
b) Satweight behavior in entry 6.3

Figure 6. 6 favors in supporting and confirming the above statement; as an elaboration of the weight plot, we observe that the number of possible assignments has increased from 3 in the previous settings to 7 assignments in the current setting.
However, when looking at the SatWeight, which is enforced by the constraints semantic, constraints implication on P is highly noticeable. Relatively, it affects the problem satisfiability behavior, not only by dropping the probabilistic weight from 0.3333 at max (as per entry 6.2) to 0.0634 at max (as per current entry), but also by mounting four unsatisfiable assignments.

In Figure 6. 6, we observe logarithmic decay in the SatWeight value while changing number of associated variables in the world of interest and when none of the variables entangled with a constraint. The highest SatWeight value has been detected when the number of variables where minimal.

The logarithmic response would change when variables exhibit association with crosstree constraints; such that number of constraints is increasing.

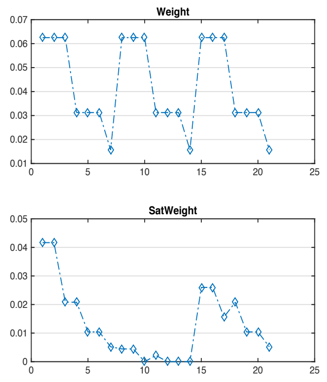
Moreover, Figure 6. 6(b) demonstrates the linear relationship between number of variables, SatWeight value, and the number of involved constraints in any world. On one hand, the SatWeight has the highest value when the number of constraints is minimal. Oppositely, on the other hand, the SatWeight becomes zero when the number of constraints reaches its maximum.

Entry 6.4. To have a better understanding of the constraints implications; we will consider a new combination in which;

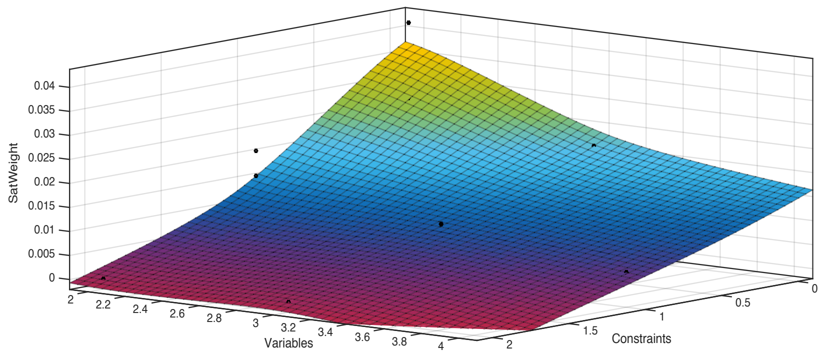
$$\lambda = \left\langle \lambda_{\bullet}\left(\lambda_{+}, \lambda_{\oplus}\right)\right\rangle$$

Figure 6. 7 demonstrates problem *P* behavior when inducing the new context combination.

Due to the new context semantics, *P* exhibits higher degree of variability. Hence, number of possible assignments has increased significantly from 7 (as per entry 6.3) to 21 assignments with highest weight of 0.0625 and the minimal is 0.01562.



(a)



(b)

Figure 6. 7 a) difference between satweight and weight in entry 6.4 b) Satweight behavior in entry 6.4

When taking , 4 out of the 21 assignments would exhibit unsatisfiable assignments. Noting that the difference among world's weight and SatWeight is not as high as in the other combinations, the highest obtained SatWeight assignment is 0.041666.

Figure 6. 7(b) displays the change of SatWeight values with respect to the variation in number of variables and involved constraints. As evident from the aforementioned cases, SatWeight tend to exhibit logarithmic response as the number of variables changes.

Adding up to the interpretations, an interesting observation displays the decay of the usual logarithmic response SatWeight and number of variables becomes more linear (at worlds with low SatWeight). This is because of the linear increment of the number involved constraints. When number of constraints is maximal, SatWeight value dropped to zero.

Furthermore, the plot also evinces that SatWeight value decreases in a linear relationship when the number of the constraints increasing linearly.

Entry 6.5. Now we are going to increase the degree of the variability of *P* by considering a new combination; such that;

$$\lambda = \left\langle \lambda_{\bullet}(\lambda_{+}, \lambda_{+}) \right\rangle$$

Semantically speaking, higher degree of variability implies more possible assignments, which is our case where the numbers of assignments have increased from 21 (as per entry 6.4) to 49 possible assignments.

According to Figure 6. 8(a), when considering the crosstree constraint, 16 out of the 49 assignments will be unsatisfiable.

Moreover, we discern that the highest SatWeight assignment value is 0.0803, which is significantly smaller than the equivalent weight value of the same world which was equal 0.25.

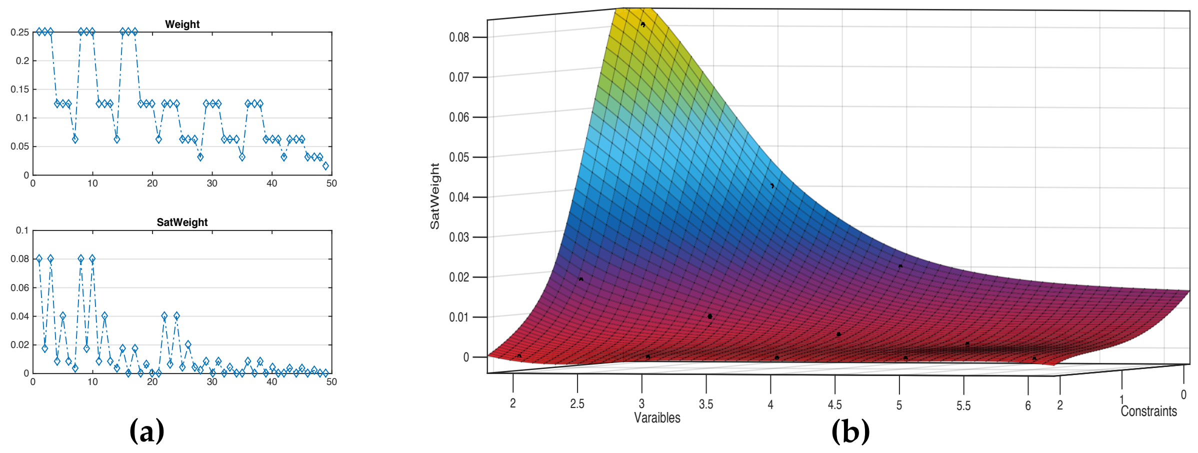


Figure 6. 8 a) difference between satweight and weight in entry 6.5
b) Satweight behavior in entry 6.5

Figure 6. 8(b) provides clear evidence how SatWeight responds logarithmically to increase of the number of involved variables.

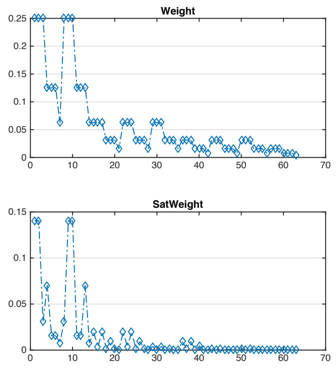
As a result of the strong logarithmic characteristics of *P*, the dominancy of the logarithmic behavior extend its effect on the linear response between crosstree constraints and SatWeight imposing some logarithmic decay along the linear response, Figure 6. 8(b) marks that crosstree constraints tend to have semi logarithmic behavior with Satweight value, however the slope of the decay is still relatively high.

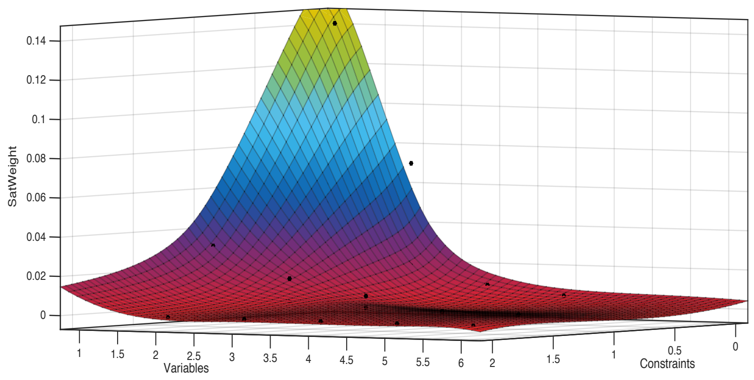
Generally, we can establish that *P* tends to have a more logarithmic reasoning behavior, when the degree of variability increases.

Entry 6.6. To have a better understanding of the previous observation, we are now going to increase the degree of variability by considering the following combination;

$$\lambda = \left\langle \lambda_{+}\left(\lambda_{+},\lambda_{+}\right)\right\rangle$$

In this combination, P obtains its highest degree of variability with 63 possible truth assignments.
As shown in Figure 6. 9(a), the highest obtained weight value is 0.25, whereas highest assignment SatWeight value is 0.140625. Moreover, 16 assignments have zero SatWeight value.





(a) **(b)**

Figure 6. 9 a) difference between satweight and weight in entry 6.6
b) Satweight behavior in entry 6.6

Figure 6. 9(b) confirms our previous observation, in which the increment in the degree of variability enforces more logarithmic behavior of the satisfiability process.
In this problem setting, we observe the logarithmic response between the number of constraints and SatWeight, inducing more logarithmic response between the number of constraints and the number of variables.

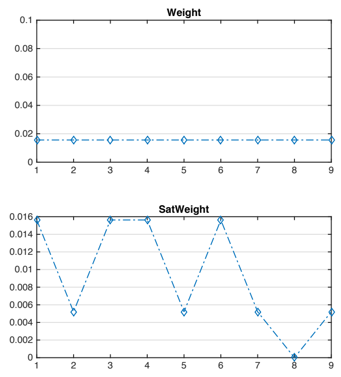
Entry 6.7. Now we are going to consider a new combination whereas;

$$\lambda = \left\langle \lambda_{\bullet}\left(\lambda_{\oplus}, \lambda_{\oplus}\right)\right\rangle$$

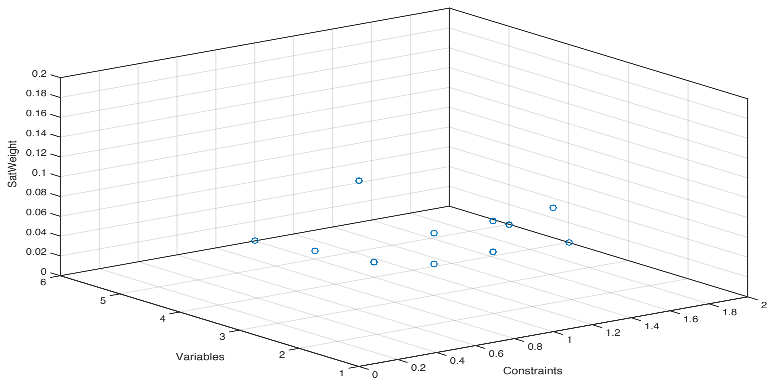
According to Figure 6. 10, the new combination leaves us with nine possible assignments with equal probable weight value of 0.015625. After considering the crosstree constraints implication, one of these assignments will be unsatisfiable with zero SatWeight value.

Moreover, we observe that some assignments maintained the same probabilistic weight before and after considering the crosstree constraints. Due to the high independency level among variables (due to the problem dependency semantic) constraints didn't take affect on many possible worlds, achieving high Satweight value comparing with other combinations.

This observation is compatible with assumption 6.1 and confirms that when imposing Bayesian Exclusive Disjunction dependency context; crosstree constraints implication would be reduced accordingly.



(a)



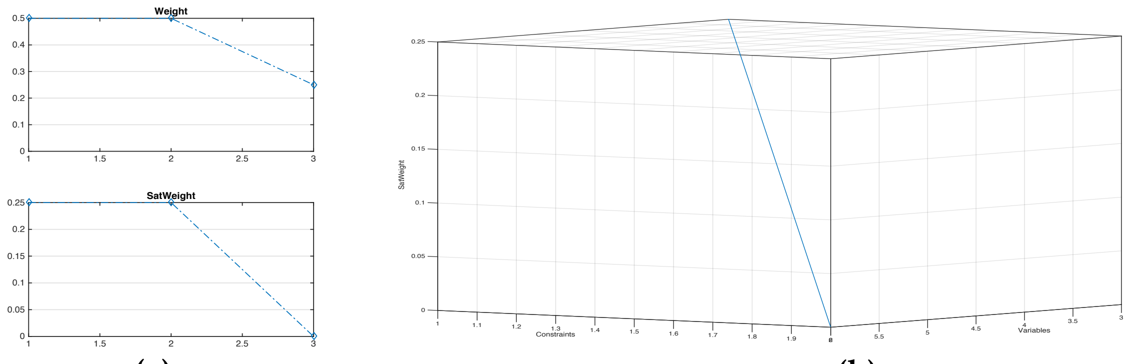
(b)

Figure 6. 10 a) difference between satweight and weight in entry 6.7
b) Satweight behavior in entry 6.7

Entry 6.8. Now we will consider a new combination whereas;

$$\lambda = \left\langle \lambda_{+}\left(\lambda_{\bullet}, \lambda_{\bullet}\right)\right\rangle$$

Due to the low degree of variability the problem qualified only three assignments with maximum weight of 0.5. When considering the crosstree constraints implication, one of these assignments exhibit unsatisfiable assignment see Figure 6. 11(b).



(a) **(b)**

Figure 6. 11 a) difference between satweight and weight in entry 6.8
b) Satweight behavior in entry 6.8

As a result of the low degree of variability, the relationship among number of variables, constraints and SatWeight tend to be linear as demonstrated in Figure 6. 11(b).

Accordingly, due to the low degree of variability, *P* would be highly influenced of the linear correlation Satweight and number of involved constraints, extending this behavior on overall problem behavior. Whereas, the expected logarithmic response between number of variables and Satweight is weakly characterized as a result of the lower degree of variability.

Entry 6.9. A higher degree of variability can be achieved when considering the following combination;

$$\lambda = \left\langle \lambda_{+}\left(\lambda_{\bullet}, \lambda_{\oplus}\right)\right\rangle$$

7 possible assignments are produced by this combination; highest weight value achieved is 0.0625. After considering the constraints implication, one assignment would be disqualified as valid assignments as shown in Figure 6. 12(a).

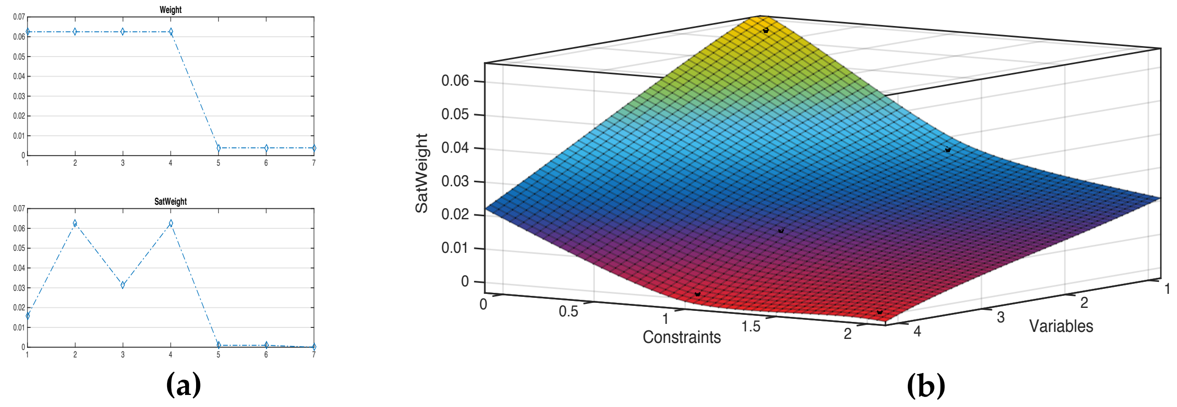


Figure 6. 12 a) difference between satweight and weight in entry 6.9
b) Satweight behavior in entry 6.9

A very interesting observation to be noted is, when exclusively examine truth assignments produced from context of higher independency level (in this case exclusive disjunction), the obtained SatWeight values are either remain the same as its correspondent weigh value or have been decreases in a reasonable amount. Whereas, assignments produced via context with higher dependency level (Conjunction dependency context) dropped its SatWeight significantly (almost forth of its correspondent weight value).

Figure 6. 12(b) also confirms this observation, when four variables are present, which are a result of incorporating the conjunction context to the inclusive disjunction context, SatWeight curve decay linearly due to the prominence of the low degree variability context behavior.

On the same note, if the number of variables was one, which only can be achieved when we only consider the higher degree of variability exclusive disjunction, SatWeight decay tend to be corresponding logarithmically as observed in Figure 6. 12(b). Still, the number of crosstree constraint has linear effect on the SatWeight value.

Entry 6.10. Entry 10 shows a Higher degree of variability to be obtained by considering the following combination such that;

$$\lambda = \left\langle \lambda_{+} \left(\lambda_{\bullet}, \lambda_{+} \right) \right\rangle$$

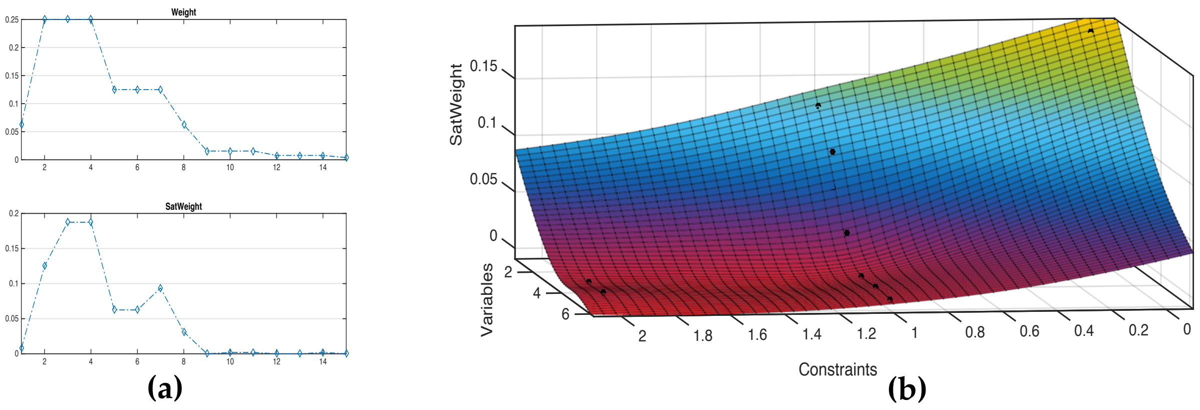


Figure 6. 13 a) difference between satweight and weight in entry 6.10
b) Satweight behavior in entry 6.10

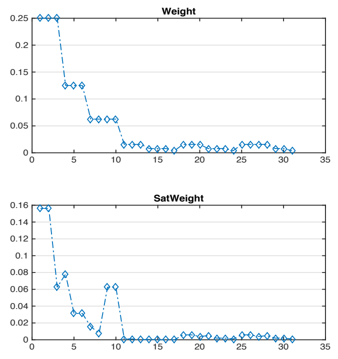
Figure 6. 13 serves as a confirmation to the previous observations, in such a way it is discerned that the difference between weight and SatWeight values is bigger for the assignments coming from context with lower degree of variability, when judged against the assignments coming from context with higher degree of variability.

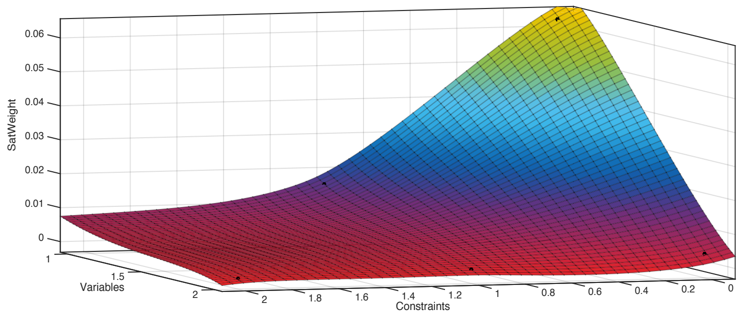
In Figure 6. 13b, we remark upon the logarithmic response between SatWeight and number of variables. When fixing number of constraint to 1, we encounter 8 possible assignments in which 4 of these assignments are derived from context with higher degree of variability (Bayesian disjunction), while the other 4 assignments are composed from both contexts.

Patently, the higher degree of variability context is dominant on this sector and as a result, the logarithmic relationship with satweight value is more visible at this sector, unlike other sectors where's the logarithmic behavior is not as visible.

Entry 6.11. In this entry we present the next combination in which we try to increase the degree of variability such that;

$$\lambda = \left\langle \lambda_{+} \left(\lambda_{+}, \lambda_{\oplus} \right) \right\rangle$$





(a) **(b)**

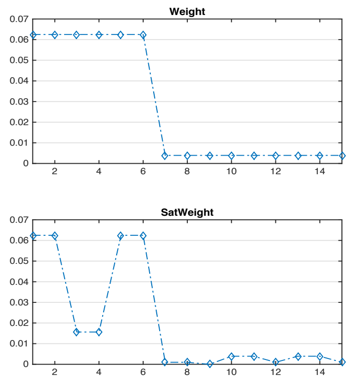
Figure 6. 14 a) difference between satweight and weight in entry 6.11
b) Satweight behavior in entry 6.11

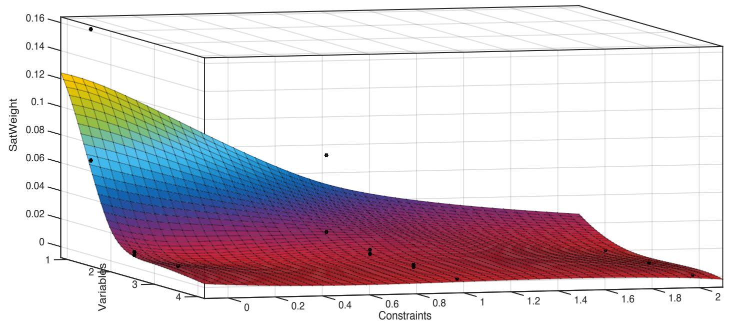
As manifestly marked from Figure 6. 14, and due to the achieved high degree of variability coupled with the degree of independency, SatWeight tends to respond more logarithmically with the change in number constraints and number of involved variables.

Also, the effect of the constraints have been reduced in which only 4 out of 31 assignments are classified as invalid assignments, which is due to the independency nature of the imposed exclusive disjunction.

Entry 6.12. In sequence of formulating a clearer understanding, entry 12 impose more independence contexts whereas;

$$\lambda = \left\langle \lambda_{+} (\lambda_{\oplus}, \lambda_{\oplus}) \right\rangle$$





(a) **(b)**

Figure 6. 15 a) difference between satweight and weight in entry 6.12
b) Satweight behavior in entry 6.12

Figure 6. 15 presents an interesting observation, due to nature of the exclusive disjunction context; the independency level in this problem has been increased. Consequently, this succeeded to reduce the degree of variability.

However, on the other hand, it also has a direct effect on subsidizing the constraints implication on SatWeight value.

From Figure 6. 15(a), we only detect one assignment out of 15 assignments, thus disqualify the crosstree constraints requirement.

In addition, when referring to Figure 6. 15(b), we distinguish the logarithmic behavior between number of constraints and SatWeight value. Besides, as a result of decreasing the degree of variability, the relationship between number of variables and SatWeight conspicuously has become more linear.

6.7. Discussion

From the analysis above we were able to determine some observations as discussed below:

In any uncertain CSP, interaction between variables is limited by the involved constraints and existing dependency context. Different dependency context allow different interface of the problem. In addition to that, each dependency context has different degree of variability, resulting higher or lower set of truth assignments. When two or more dependency contexts aggregate together, the crosstree constraints effect on the satisfiability of the problem become more prominent. Different techniques can be imposed to minimize the constraints effect and achieve optimized satisfiable problem.

Moreover, It's also important to notice that the arrangement of dependency context combinations has a crucial effect on the system's satisfiability behavior. Different arrangements of dependency contexts result in different degree of variability, which in its turn has a direct effect on the constraints impact; as have been shown above and demonstrated in Figure 6. 16.

Considering these observations, we argue that not only we achieve better understanding of the problem reasoning behavior, but at same time we introduce some techniques to optimize the reasoning process and reduce complexity time during the modeling stage and even after.

The analysis above evidenced that in any Uncertain CSP, SatWeight value responds linearly with the change of number of involved crosstree constraints. Besides, the problem degree of variability tends to have logarithmic effect on the SatWeight value. Higher degree of variability means more logarithmic response, and vice versa.

	Degree of Variability	Independence Level	Constraints Implications	SatWeight	Problem Behaviour	
					Variables vs SatWeight	Constraints vs SatWeight
$\lambda_{+}\lambda_{+}$	+	—	+	—	Logarithmic	Exponential
$\lambda_{+}\lambda_{\bullet}$	—	—	+	—	Logarithmic	Exponential
$\lambda_{\bullet}\lambda_{\bullet}$	Min	Min	Max	Zero	unsatisfiable	unsatisfiable
$\lambda_{\oplus}\lambda_{\bullet}$	—	—	+	—	Linear	Exponential
$\lambda_{\oplus}\lambda_{+}$	±	±	—	+	Linear	Linear
$\lambda_{\oplus}\lambda_{\oplus}$	—	+	—	+	Logarithmic	Linear
$\lambda_{+}\lambda_{+},\lambda_{+},\lambda_{+}$	+	—	±	±	Logarithmic	Logarithmic
$\lambda_{+}\lambda_{\bullet},\lambda_{+},\lambda_{\bullet}$	—	—	±	±	Linear	Linear
$\lambda_{\bullet}\lambda_{\bullet},\lambda_{\bullet},\lambda_{\bullet}$	—	—	±	—	Linear	Exponential
$\lambda_{\oplus}\lambda_{\bullet},\lambda_{\oplus},\lambda_{\bullet}$	—	—	—	+	Linear	Linear
$\lambda_{\oplus}\lambda_{+},\lambda_{\oplus},\lambda_{+}$	+	+	—	+	Linear	Logarithmic
$\lambda_{\oplus}\lambda_{\oplus},\lambda_{\oplus},\lambda_{\oplus}$	±	+	—	+	Logarithmic	Linear

Figure 6. 16 Uncertain CSP behavior under different dependency configurations

Manifestly, problems with low degree of variability and high number of constraints have an obvious linear reasoning behavior. This results in the likelihood of the SatWeight value to drop faster and the chances of having unsatisfiable assignments to be relatively high. On the contrary, problems with high degree of variability and low number of constraints will tend to have logarithmic reasoning behavior with higher success chances.

Having that said, we argue that we can optimize the satisfiability outcome by inducing higher degree of variability in the problem, which will result to increase the logarithmic behavior in the problem and enforce this change on the linear response between number of constraints and Satweight value, making it more logarithmic as have been revealed in the analysis above.

Introducing a new degree of variability coupled with degree of independency in the problem will reduce the crosstree constraints effect as well as formulating a logarithmic effect on the SatWeight value.

In this manner, the degree of variability can be increased by either introducing disjunction dependency or exclusive disjunction dependency into the problem.

When engaging exclusive disjunction dependency to the problem, a significant decrease to the crosstree constraints effect will take place , due to the independency nature, will exist, and the change between weight and SatWeight value would be minimal and sometimes zero (see entry 6,7 and 6.12). However, the downfall would be arising as a reduction in number of possible assignments.

Moreover, there is a great possibility to reduce the effect of crosstree constraints and make it even logarithmic by inducing Disjunction dependency to the problem (for instance see entry 6.4, 6.5 and 6.11). Likewise, we will allow SatWeight to respond logarithmically with number of constraints and number of variables in addition of having more possible assignments. Conversely, when speaking of its downfall, the change between weight and SatWeight will be higher in comparison to the previous case as evident throughout the whole experiment.

6.5. Summary

In this chapter, we were able to extend our work by introducing a new reasoning mechanism, tailored for the developed BBFM.

We started by highlighting *Uncertain CSP* characteristics. Thereafter, an extensive elaboration of the developed method had been discussed thoroughly.

Reasoning in SPL is one of the most expensive challenges. When it comes to constraints satisfaction, reasoning might arise NP-complete problem.

In our approach, we first extract the problem dependency context; to specify its truth domain. The obtained truth domains are, thereafter, marginally factorized with the truth assignment of the involved crosstree constraints. While taking into account the predefined uncertainty measure; the developed algorithm finally conclude the valid assignments that satisfy the existing dependency, constraints semantic and any observations. Moreover, each satisfiable assignment is to be coupled with uncertainty measure in accordance with the embedded observation.

To enhance the algorithm performance, we introduced several techniques aiding in reducing the problem space size, and the search for optimal observations.

Finally, we validated the developed algorithm through out set of experiments, each representing a new sub-model with different dependencies, variability, and complexity. The obtained results demonstrated that, the developed algorithm is reliable and efficiently help the designer to understand the problem behavior under different sittings.

Part VI

Conclusion

Chapter 7

Conclusion

Important decisions are made based on Feature Models. After decades of research in SPLE, different approaches have emerged for creating and managing variability models in SPLE (see chapter Managing Variabilities). Nevertheless, all proposed techniques exhibit some sort of limitations concerning dependencies' semantic and non-direct interaction among model parameters. (Apel et al., 2013a; Benavides et al., 2010), and consequently, provide unclear support for the reasoning process thereafter.

To capture the actual semantic existing in the feature model, one should consider the valid implication for each parameter on the overall model. Moreover, when defining each parameter real implication, we will be able to anticipate the model outcome and features behavior in the latter stages. By underlining features behavior, we would be able to quantify the model possible functionalities in the early design stage.

This thesis presented a novel approach to quantify features implications and anticipate the truth assignment for these features in all possible products configurations. After assigning truth weights for all features, we use these measurements to enhance the graphical representation of the introduced feature model by integrating the developed model with use of colour, such that each colour emphasize different truth value.

The predefined measurements are used later during the reasoning process, in which we developed a mathematical reasoner to satisfy about the model constraints throughout a consideration of the given probabilistic weight for each parameter.

First, we started this research by conducting a Systematic Literature Review identifying the current modeling techniques used in SPLE. Next, we evaluate each technique to highlight its qualities and determine its usage and limitation as been instituted in the literature. After defining each modeling technique and highlight its shortcomings, we drafted a comparative study to conclude the pros and cons of each used model.

Based on the findings of the previous study, we were able to identify the Knowledge Gap as have been discussed in section 4.4.

To overcome the knowledge gap, we integrated the use of Bayesian Belief Networks BBN with Feature Model FM, producing a probabilistic feature model capable of capturing the introduced dependency semantic found in FM.

The developed Bayesian Belief Feature Model BBFM is a mathematical framework to model and manage variability model in SPLE by quantifying the truth uncertainty of all model parameters. BBFM support a set of theorems and mathematical notions, in which we enable the model designer to assign a probabilistic weight for each feature, indicating the truth assumptions and the occurrence likelihood for all features in BBFM. While quantifying the truth probabilistic weight for models' parameters, we can anticipate the truth flow throughout the model parameters.

Afterwards, we extent these analysis to compute the truth assumption of model dependency contexts by anticipating the probability of obtaining valid context in which the dependency semantic is met and all involved constraints are satisfied.

The obtained analysis formulates the model belief network, which mainly originated from the truth assignment of core features and bounded by the semantic of crosstree constraints. Thus are used to colour the BBFM in shades of gray fashion, where each model parameters will be assigned a colour shade in corresponding with its truth assumption. This allows better visualization of the model belief base, while emphasizing the truth flow, features implications and interaction throughout the model.

Successively, we demonstrated how the model belief could be changed subjectively when deciding to include some model parameters in the belief base and trace the anticipated changes graphically in the same exact fashion discussed above.

After developing BBFM, we continued this work by creating a mathematical reasoner to tackle the problem of constraints satisfaction in SPLE. The developed reasoner is a *state of art* reasoning technique; the designed approach is a result of integrating the traditional constraints satisfaction problem and reasoning under uncertainty theorems. Nevertheless, it was specially designed and tailored to reason about BBFM. We were able to mathematically prove the validity of the developed approach and demonstrate its functionality through out set of examples. When reasoning using the developed algorithm, we will be able to anticipate the probability of having a satisfiable assignment for any set features in BBFM. Moreover, the any unsatisfiablity would be detected when using the developed approach.

To testify the developed approach, we conduct a set of experiments compiling different satisfaction problems with different attributes. The outcome of the experiments proved the model functionality and provided insightful observations about the satisfaction problem behavior under different settings.

In the next section, we highlight the directions of our planned future work.

7.1. Future Work

As future work, we will continue to extend this work by creating a MatLab Toolbox to automate the introduced analysis. For example, after defining the problem knowledge domain, we translate the intended semantics into blind BBFM, in which we don't quantify the model parameters probability weight. Afterward, we define the model belief base, then use the existing semantics and the belief base to derive our calculation using the designed toolbox. The toolbox is a mathematical implementation of the aforementioned theorems and techniques, it consist of all the introduced equations and theorems.

We also encourage the researchers to develop variability management tool, such that we will be able to manage the modeling problem by considering the probabilistic weight for all model parameters.

The developed tool must be a comprehensive management tool, in which we cover all different key aspects of SPL ranging from the model design to products configuration. This enables the user to enhance the design accordingly, by either providing subjective truth assignments for some model parameters or highlighting inconsistencies and dead features when quantifying the uncertainty measurements.

Different theorems can be employed in this framework to quantify new attributes of any software product line. Such as the use information theory to project the model entropy into countable information whereas, the key input is the computed uncertainty measures.

In addition, we would extend this work by providing a mathematical approach to estimate the number of possible valid products under different settings (observation and contexts).

Ultimately, our goal is to inspire other researchers to incorporate this work with other automated analysis, whereas the uncertainty nature of model parameters is a key attribute and should be considered in all stages. Moreover we urge to design a comprehensive tool capable of managing the variability, modeling and reasoning SPL while considering the degree of uncertainty arisen in each problem.

References

Adamides, E. D. (1996). Responsibility-based manufacturing. *The International Journal of Advanced Manufacturing Technology, 11*(6), 439-448.

Åhlström, P., & Westbrook, R. (1999). Implications of mass customization for operations management: an exploratory survey. *International Journal of Operations & Production Management, 19*(3), 262-275.

Ahmed, F., & Capretz, L. F. (2011). An architecture process maturity model of software product line engineering. *Innovations in Systems and Software Engineering, 7*(3), 191-207.

Alencar, P. (2012). Handbook of Research on Mobile Software Engineering: Design, Implementation, and Emergent Applications: Design, Implementation, and Emergent Applications, IGI Global.

Alves, V., et al. (2006). Refactoring product lines. Proceedings of the 5th international conference on Generative programming and component engineering, ACM.

Amine, A., Mohamed, O. A., & Bellatreche, L. (2013). *Modeling Approaches and Algorithms for Advanced Computer Applications* (Vol. 488): Springer.

Apel, S., Batory, D., Kästner, C., & Saake, G. (2013a). A Development Process for Feature-Oriented Product Lines *Feature-Oriented Software Product Lines* (pp. 17-44): Springer.

Apel, S., Batory, D., Kästner, C., & Saake, G. (2013b). *Feature-oriented software product lines: concepts and implementation*: Springer Science & Business Media.

Arcaini, P., Gargantini, A., & Vavassori, P. (2015). *Generating tests for detecting faults in feature models.* Paper presented at the Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference.

Ardito, C., Barricelli, B. R., Buono, P., Costabile, M. F., Lanzilotti, R., Piccinno, A., & Valtolina, S. (2011). An ontology-based approach to product customization *End-User Development* (pp. 92-106): Springer.

Atkinson, C., Bayer, J., & Muthig, D. (2000). Component-based product line development: the KobrA approach *Software Product Lines* (pp. 289-309): Springer.

Bachmann, F., & Clements, P. C. (2005). *Variability in software product lines*.

Bak, K. (2013). Modeling and analysis of software product line variability in Clafer.

Batory, D. (2005). *Feature models, grammars, and propositional formulas*: Springer.

Batory, D., Benavides, D., & Ruiz-Cortes, A. (2006). Automated analysis of feature models: challenges ahead. *Communications of the ACM, 49*(12), 45-47.

Benavides, D., Cortés, A. R., Trinidad, P., & Segura, S. (2006). *A Survey on the Automated Analyses of Feature Models.* Paper presented at the JISBD.

Benavides, D., Ruiz-Cortés, A., Corchuelo, R., & Martín-Díaz, O. (2004). *Spl needs an automatic holistic model for software reasoning with feature models.* Paper presented at the International Workshop on Requirements Reuse in System Family Engineering.

Benavides, D., Segura, S., & Ruiz-Cortés, A. (2009). Automated analysis of feature models: A detailed literature review. *Techn. Ber. ISA-09-TR-04. Seville, Spain: Applied Software Engineering Research Group, University of Seville*.

Benavides, D., Segura, S., & Ruiz-Cortés, A. (2010). Automated analysis of feature models 20 years later: A literature review. *Information Systems, 35*(6), 615-636.

Benavides, D., Segura, S., Trinidad, P., & Cortés, A. R. (2007). FAMA: Tooling a Framework for the Automated Analysis of Feature Models. *VaMoS, 2007*, 01.

Benavides, D., Segura, S., Trinidad, P., & Ruiz-Cortés, A. (2006a). A first step towards a framework for the automated analysis of feature models. *Proc. Managing Variability for Software Product Lines: Working With Variability Mechanisms*, 39-47.

Benavides, D., Segura, S., Trinidad, P., & Ruiz-Cortés, A. (2006b). Using Java CSP solvers in the automated analyses of feature models *Generative and Transformational Techniques in Software Engineering* (pp. 399-408): Springer.

Berg, K., Bishop, J., & Muthig, D. (2005). *Tracing software product line variability: from problem to solution space.* Paper presented at the Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries.

Berg, K., & Muthig, D. (2005). A critical analysis of using feature models for variability management. *Submitted to SPLC-Europe*.

Berger, T. (2013). *Variability Modeling in the Real-An Empirical Journey from Software Product Lines to Software Ecosystems.*

Beuche, D., et al. (2004). "Variability management with feature models." Science of Computer Programming 53(3): 333-352.

Beuche, D., & Dalgarno, M. (2007). Software product line engineering with feature models. *Overload Journal, 78*, 5-8.

Bézivin, J. (2001). *From object composition to model transformation with the MDA.* Paper presented at the tools.

Biglever software inc. Product Line Engineering Solutions for Systems and Software. Retrieved from http://www.biglever.com/extras/BigLever_Solution_Brochure.pdf

Boehm, B. W. (1981). *Software engineering economics* (Vol. 197): Prentice-hall Englewood Cliffs (NJ).

Bontemps, Y., et al. (2005). Generic Semantics of Feature Diagrams Variants. FIW.

Bosch, J., & Bosch-Sijtsema, P. (2010). From integration to composition: On the impact of software product lines, global development and ecosystems. *Journal of Systems and Software, 83*(1), 67-76.

Bosch, J., Florijn, G., Greefhorst, D., Kuusela, J., Obbink, J. H., & Pohl, K. (2001). Variability issues in software product lines *Software Product-Family Engineering* (pp. 13-21): Springer.

Chastek, G., et al. (2001). Product line analysis: a practical introduction, DTIC Document.

Chen, K., Zhang, W., Zhao, H., & Mei, H. (2005). *An approach to constructing feature models based on requirements clustering.* Paper presented at the Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference.

Chen, L., Ali Babar, M., & Ali, N. (2009). *Variability management in software product lines: a systematic review.* Paper presented at the Proceedings of the 13th International Software Product Line Conference.

Chen, P. P.-S. (1976). The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS), 1*(1), 9-36.

Classen, A., Heymans, P., & Schobbens, P.-Y. (2008). What's in a feature: A requirements engineering perspective *Fundamental Approaches to Software Engineering* (pp. 16-30): Springer.

Clements, P. C. (2002). *Software architecture in practice.* Software Engineering Institute.

Clements, P. C., Cohen, S., Donohoe, P., & Northrop, L. (2001). Control channel toolkit: a software product line case study.

Clements, P. and L. Northrop (2015). Software product lines, Addison-Wesley.

Cohen, S. (2003). *Predicting when product line investment pays*.

Cook, S. A. (1971). *The complexity of theorem-proving procedures.* Paper presented at the Proceedings of the third annual ACM symposium on Theory of computing.

Coplien, J., Hoffman, D., & Weiss, D. (1998). Commonality and variability in software engineering. *Software, IEEE, 15*(6), 37-45.

Crnkovic, I., & Larsson, M. P. H. (2002). *Building reliable component-based software systems*: Artech House.

Cuevas, D. B. (2007). *On the Automated Analysis of Software Product Lines Using Feature Models.* Citeseer.

Czarnecki, K., & Eisenecker, U. W. (2000). Generative programming. *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen*, 15.

Czarnecki, K., Grünbacher, P., Rabiser, R., Schmid, K., & Wąsowski, A. (2012). *Cool features and tough decisions: a comparison of variability modeling approaches.* Paper presented at the Proceedings of the sixth international workshop on variability modeling of software-intensive systems.

Czarnecki, K., & Helsen, S. (2003). *Classification of model transformation approaches.* Paper presented at the Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture.

Czarnecki, K., Helsen, S., & Eisenecker, U. (2004). Staged configuration using feature models *Software Product Lines* (pp. 266-283): Springer.

Czarnecki, K., Helsen, S., & Eisenecker, U. (2005). Formalizing cardinality‐based feature models and their specialization. *Software process: Improvement and practice, 10*(1), 7-29.

Czarnecki, K., & Kim, C. H. P. (2005). *Cardinality-based feature modeling and constraints: A progress report.* Paper presented at the International Workshop on Software Factories.

Czarnecki, K., & Wasowski, A. (2007). *Feature diagrams and logics: There and back again.* Paper presented at the Software Product Line Conference, 2007. SPLC 2007. 11th International.

Da Silveira, G., Borenstein, D., & Fogliatto, F. S. (2001). Mass customization: Literature review and research directions. *International journal of production economics, 72*(1), 1-13.

Davis, S. M. (1989). From “future perfect”: Mass customizing. *Planning review, 17*(2), 16-21.

Dermeval, D., et al. (2015). "Ontology-based feature modeling: An empirical study in changing scenarios." Expert Systems with Applications 42(11): 4950-4964.

Donohoe, P. (2012). Software product lines: Experience and research directions, Springer Science & Business Media.

Dubslaff, C., Klüppeholz, S., & Baier, C. (2014a). Probabilistic Software Product Line Model Checking.

Dubslaff, C., et al. (2014b). Probabilistic model checking for energy analysis in software product lines. Proceedings of the 13th international conference on Modularity, ACM.

Dubslaff, C. (2015). Advances in Quantitative Software Product Line Analysis. Software Engineering & Management.

Eastwood, M. A. (1996). Implementing mass customization. *Computers in industry, 30*(3), 171-174.

Ebert, C., & Smouts, M. (2003). *Tricks and traps of initiating a product line concept in existing products.* Paper presented at the Proceedings of the 25th International Conference on Software Engineering.

Elfaki, A. O., et al. (2012). "Review and future directions of the automated validation in software product line engineering." Journal of Theoretical and Applied Information Technology 42(1).

Eriksson, M., & Hagglunds, A. (2003). *An Introduction to Software Product Line Development.* Paper presented at the Proceedings of Umeå's Seventh Student Conference in Computing Science.

Ferber, S., et al. (2002). Feature interaction and dependencies: Modeling features for reengineering a legacy product line. Software product lines, Springer: 235-256.

Ferré, X. and S. Vegas (1999). An evaluation of domain analysis methods. 4th CASE/IFIP8 International Workshop in Evaluation of Modeling in System Analysis and Design, Citeseer.

Gacek, C. and M. Anastasopoules (2001). Implementing product line variabilities. ACM SIGSOFT Software Engineering Notes, ACM.

Gargantini, A., & Fraser, G. (2011). Generating minimal fault detecting test suites for general boolean specifications. *Information and Software Technology, 53*(11), 1263-1273.

Gibson, J. P. (1997). *Feature Requirements Models: Understanding Interactions.* Paper presented at the FIW.

Gil, R. H. and D. F. Amorós (2009). Towards a time-efficient algorithm to calculate the total number of products of a Software Product Line. Proceedings of the 1st International Workshop on Domain Engineering, Citeseer.

Glück, R., & Lowry, M. (2005). *Generative Programming and Component Engineering: 4th International Conference, GPCE 2005, Tallinn, Estonia, September 29-October 1, 2005, Proceedings* (Vol. 3676): Springer.

Gomaa, H. (2005). *Designing software product lines with UML*: IEEE.

Griss, M. L., Favaro, J., & Alessandro, M. D. (1998). *Integrating feature modeling with the RSEB.* Paper presented at the Software Reuse, 1998. Proceedings. Fifth International Conference.

Halmans, G., & Pohl, K. (2003). Communicating the variability of a software-product family to customers. *Software and Systems Modeling, 2*(1), 15-36.

Harman, M., Jia, Y., Krinke, J., Langdon, W. B., Petke, J., & Zhang, Y. (2014). *Search based software engineering for software product line engineering: a survey and directions for future work.* Paper presented at the Proceedings of the 18th International Software Product Line Conference-Volume 1.

Hart, C. W. (1995). Mass customization: conceptual underpinnings, opportunities and limits. *International Journal of Service Industry Management, 6*(2), 36-45.

Haugen, Ø., Moller-Pedersen, B., Oldev, J., Olse, G. K., & Svendsen, A. (2008). *Adding standardized variability to domain specific languages.* Paper presented at the Software Product Line Conference, 2008. SPLC'08. 12th International.

Heuser, C. A., & Pernul, G. (2009). *Advances in Conceptual Modeling-Challenging Perspectives: ER 2009 Workshops CoMoL, ETheCoM, FP-UML, MOST-ONISW, QoIS, RIGiM, SeCoGIS, Gramado, Brazil, November 9-12, 2009, Proceedings* (Vol. 5833): Springer.

Heymans, P., Schobbens, P.-Y., Trigaux, J.-C., Bontemps, Y., Matulevicius, R., & Classen, A. (2008). Evaluating formal properties of feature diagram languages. *Software, IET, 2*(3), 281-302.

Hirsch, B., Thoben, K.-D., & Hoheisel, J. (1998). Requirements upon human competencies in globally distributed manufacturing. *Computers in industry, 36*(1), 49-54.

Hubaux, A. (2012). Feature-based configuration: Collaborative, dependable, and controlled, FUNDP.

Hubaux, A., et al. (2010). Evaluating a textual feature modelling language: four industrial case studies. Software Language Engineering, Springer: 337-356.

Hubaux, A., et al. (2010). "A Preliminary Review on the Application of Feature Diagrams in Practice." VaMoS 10: 53-59.

Hubaux, A., et al. (2008). Variability modeling challenges from the trenches of an open source product line re-engineering project. Software Product Line Conference, 2008. SPLC'08. 12th International, IEEE.

Hubaux, A., et al. (2013). "Supporting multiple perspectives in feature-based configuration." Software & Systems Modeling 12(3): 641-663.

Hutchesson, S. and J. McDermid (2013). "Trusted product lines." Information and Software Technology 55(3): 525-540.

John, I., & Muthig, D. (2002). *Tailoring use cases for product line modeling.* Paper presented at the Proceedings of the International Workshop on Requirements Engineering for product lines.

Joneja, A., & Lee, N. K. (1998). Automated configuration of parametric feeding tools for mass customization. *Computers & industrial engineering, 35*(3), 463-466.

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, A. S. (1990). *Feature-oriented domain analysis (FODA) feasibility study*.

Kang, K. C., Kim, S., Lee, J., Kim, K., Shin, E., & Huh, M. (1998). FORM: A feature-; oriented reuse method with domain-; specific reference architectures. *Annals of Software Engineering, 5*(1), 143-168.

Kay, M. J. (1993). Making mass customization happen: Lessons for implementation. *Planning review, 21*(4), 14-18.

Keele, S. (2007). Guidelines for performing systematic literature reviews in software engineering *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*.

Kiniry, J. (2007). *Reasoning about feature models in higher-order logic.* Paper presented at the Proceedings of the 11th International Software Product Line Conference, SPLC'07. IEEE Computer Society.

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering–a systematic literature review. *Information and Software Technology, 51*(1), 7-15.

Knauber, P., Muthig, D., Schmid, K., & Widen, T. (2000). Applying product line concepts in small and medium-sized companies. *Software, IEEE, 17*(5), 88-95.

Kotha, S. (1995). Mass customization: implementing the emerging paradigm for competitive advantage. *Strategic Management Journal, 16*(S1), 21-42.

Kotha, S. (1996). Mass–customization: a strategy for knowledge creation and organizational learning. *International Journal of Technology Management, 11*(7-8), 846-858.

Kotler, P. (1989). From mass marketing to mass customization. *Planning review, 17*(5), 10-47.

Kullback, S. (1968). *Information theory and statistics*: Courier Corporation.

Lang, M. (2015). *Design of a Portfolio Management System for Software Line Development: Merging the Gap between Software Project and Product Management*: diplom. de.

Lau, R. S. (1995). Mass customization: the next industrial revolution. *Industrial Management; Norcross, 37*(5), 18.

Lee, K., Kang, K. C., Chae, W., & Choi, B. W. (2000). Feature-based approach to object-oriented engineering of applications for reuse. *Software-Practice and Experience, 30*(9), 1025-1046.

Lee, K., Kang, K. C., & Lee, J. (2002). Concepts and guidelines of feature modeling for product line software engineering *Software Reuse: Methods, Techniques, and Tools* (pp. 62-77): Springer.

Macala, R. R., Stuckey Jr, L. D., & Gross, D. C. (1996). Managing domain-specific, product-line development. *Software, IEEE, 13*(3), 57-67.

Mannion, M. (2002). Using first-order logic for product line model validation *Software Product Lines* (pp. 176-187): Springer.

Männistö, T., & Bosch, J. (2004). Workshop on Software Variability Management for Product Derivation—Towards Tool Support *Software Product Lines* (pp. 331-331): Springer.

Mazo, R., & Salinesi, C. (2008). Methods, techniques and tools for product line model verification.

McGregor, J. D. (2005). "Preparing for automated derivation of products in a software product line."

Mendonça, M. (2009). *Efficient reasoning techniques for large scale feature models.* University of Waterloo.

Mendonca, M., Branco, M., & Cowan, D. (2009). *SPLOT: software product lines online tools.* Paper presented at the Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications.

Mendonca, M., Wąsowski, A., & Czarnecki, K. (2009). *SAT-based analysis of feature models is easy.* Paper presented at the Proceedings of the 13th International Software Product Line Conference.

Metzger, A., Pohl, K., Heymans, P., Schobbens, P.-Y., & Saval, G. (2007). *Disambiguating the documentation of variability in software product lines: A separation of concerns, formalization and automated analysis.* Paper presented at the Requirements Engineering Conference, 2007. RE'07. 15th IEEE International.

Moody, D. L. (2009). The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. *Software Engineering, IEEE Transactions on, 35*(6), 756-779.

Muschevici, R., Proença, J., & Clarke, D. (2015). Feature Nets: behavioural modelling of software product lines. *Software & Systems Modeling*, 1-26.

Northrop, L., & Clements, P. (2001). Software product lines. *URL http://www. sei. cmu. edu/library/assets/Philips, 4*(05).

Northrop, L. M. (2006). *Software product lines: reuse that makes business sense.* Paper presented at the Software Engineering Conference, 2006. Australian.

Northrop, L. M., et al. . A Framework for Software Product Line Practice, Version 5.0 *Software Engineering Institute, PLPI.* Retrieved from http://www.sei.cmu.edu/productlines/frame_report/index.html

Perez-Morago, H., Heradio, R., Fernandez-Amoros, D., Bean, R., & Cerrada, C. (2015). Efficient Identification of Core and Dead Features in Variability Models. *Access, IEEE, 3*, 2333-2340.

PINE, J., VICTOR, B., & BOYTON, A. (1993). Making mass customization work. . *Harvard Business Review., Vol. 71*(no. 5,), p. 108-111.

Pleuss, A., Botterweck, G., Dhungana, D., Polzer, A., & Kowalewski, S. (2010). *Featureoriented modelling of product line evolution*.

Pohl, K., Böckle, G., & van Der Linden, F. J. (2005). *Software product line engineering: foundations, principles and techniques*: Springer Science & Business Media.

Pohl, R., Lauenroth, K., & Pohl, K. (2011). *A performance comparison of contemporary algorithmic approaches for automated analysis operations on feature models.* Paper presented at the Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering.

Riebisch, M. (2003). Towards a more precise definition of feature models. *Modelling Variability for Object-Oriented Product Lines*, 64-76.

Rincón, L., Giraldo, G., Mazo, R., Salinesi, C., & Diaz, D. (2015). Method to identify corrections of defects on product line models. *Electronic Notes in Theoretical Computer Science, 314*, 61-81.

Ross, A. (1996). Selling uniqueness. *Manufacturing Engineer, 75*(6), 260-263.

Schobbens, P.-Y., Heymans, P., & Trigaux, J.-C. (2006). *Feature diagrams: A survey and a formal semantics.* Paper presented at the Requirements Engineering, 14th IEEE international conference.

Shannon, C. E. (1949). Communication theory of secrecy systems*. *Bell system technical journal, 28*(4), 656-715.

Shaw, M. (1998). *Constructing Systems from Parts: What Students Should Learn about Software Architecture" and "Architectural Mismatch, Interoperability, and the Prospects for Electronic Commerce in Software Parts and Services*. Retrieved from Software Architecture and Design: Proceedings of the Joint International Computers Limited/University of Newcastle Seminar.

Shwe, M. A., Middleton, B., Heckerman, D., Henrion, M., Horvitz, E., Lehmann, H., & Cooper, G. (1991). Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. *Methods of information in Medicine, 30*(4), 241-255.

Siegmund, N., Rosenmüller, M., Kuhlemann, M., Kästner, C., Apel, S., & Saake, G. (2012). SPL Conqueror: Toward optimization of non-functional properties in software product lines. *Software Quality Journal, 20*(3-4), 487-517.

Sinnema, M., & Deelstra, S. (2008). Industrial validation of COVAMOF. *Journal of Systems and Software, 81*(4), 584-600.

Sun, J., Zhang, H., Fang, Y., & Wang, H. (2005). *Formal semantics and verification for feature modeling*. Paper presented at the Engineering of Complex Computer Systems, 2005. ICECCS 2005. Proceedings. 10th IEEE International Conference

Svahnberg, M., Van Gurp, J., & Bosch, J. (2005). A taxonomy of variability realization techniques. *Software: Practice and Experience, 35*(8), 705-754.

Thao, C. (2012). A configuration management system for software product lines.

Thiel, S., & Hein, A. (2002). Modeling and using product line variability in automotive systems. *IEEE software, 19*(4), 66.

Thüm, T., Apel, S., Kästner, C., Kuhlemann, M., Schaefer, I., & Saake, G. (2012). Analysis strategies for software product lines. *School of Computer Science, University of Magdeburg, Tech. Rep. FIN-004-2012*.

Thüm, T., Batory, D., & Kästner, C. (2009). *Reasoning about edits to feature models.* Paper presented at the Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference.

Tsang, E. (2014). *Foundations of Constraint Satisfaction: The Classic Text*: BoD–Books on Demand.

Van der Linden, F. J., Schmid, K., & Rommes, E. (2007). *Software product lines in action: the best industrial practice in product line engineering*: Springer Science & Business Media.

Van Gurp, J., Bosch, J., & Svahnberg, M. (2001). *On the notion of variability in software product lines.* Paper presented at the Software Architecture, 2001. Proceedings. Working IEEE/IFIP Conference.

Wang, H., Li, Y. F., Sun, J., Zhang, H., & Pan, J. (2005). *A semantic web approach to feature modeling and verification.* Paper presented at the Workshop on Semantic Web Enabled Software Engineering (SWESE'05).

White, J., Benavides, D., Schmidt, D. C., Trinidad, P., Dougherty, B., & Ruiz-Cortes, A. (2010). Automated diagnosis of feature model configurations. *Journal of Systems and Software, 83*(7), 1094-1107.

White, J., Dougherty, B., Schmidt, D. C., & Benavides, D. (2009). *Automated reasoning for multi-step feature model configuration problems.* Paper presented at the Proceedings of the 13th International Software Product Line Conference.

White, J., Galindo, J. A., Saxena, T., Dougherty, B., Benavides, D., & Schmidt, D. C. (2014). Evolving feature model configurations in software product lines. *Journal of Systems and Software, 87*, 119-136.

White, J., Schmidt, D. C., Benavides, D., Trinidad, P., & Ruiz-Cortés, A. (2008). *Automated diagnosis of product-line configuration errors in feature models.* Paper presented at the Software Product Line Conference, 2008. SPLC'08. 12th International.

Zave, P., & Jackson, M. (1997). Four dark corners of requirements engineering. *ACM transactions on Software Engineering and Methodology (TOSEM), 6*(1), 1-30.