Universidade Federal da Bahia
Instituto de Matemática

Programa de Pós-Graduação em Ciência da Computação

VARIABILITY IMPLEMENTATION AT STACK OVERFLOW: AN EMPIRICAL STUDY

Marco Antonio Paranhos Silva

DISSERTAÇÃO DE MESTRADO

Salvador
Setembro de 2018

MARCO ANTONIO PARANHOS SILVA

VARIABILITY IMPLEMENTATION AT STACK OVERFLOW: AN EMPIRICAL STUDY

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Advisor: Eduardo Santana de Almeida

Salvador
Setembro de 2018

RESUMO

Nesta dissertação foi realizada uma investigação empírica com o objetivo de compreender como o tema "implementação da variabilidade" é discutido pela comunidade de profissionais de Tecnologia da Informação que utilizam o site Stack Overflow. Inicialmente, foi definida uma lista de 52 palavras relacionadas a este tema com base na literatura da área e na opinião de especialistas. A lista foi utilizada como vocabulário inicial para o algorítmo que implementa a Alocação de Dirichlet Latente (LDA) para encontrar os principais tópicos relacionados à implementação da variabilidade, bem como para validar as palavras da lista. Em seguida, com as palavras mais usadas nas perguntas e respostas, validadas pela LDA, as pesquisas foram realizadas no conjunto de dados recuperado do site Stack Overflow e vários estudos foram realizados para encontrar o melhor conjunto para ser analisado, considerando que algumas palavras utilizadas isoladamente recuperaram muitas questões, pois são palavras gerais com outras utilizações na área de TI. O melhor resultadofoi obtido combinando estas palavras dois a dois e, assim, 1962 perguntas foram recuperadas. Estas 1962 peerguntas foram então analisados com o objetivo de descobrir os mecanismos de implementação de variabilidades mais discutidos, menos discutidos e não discutidos; qual é o demora na resposta às questões formais sobre a implementação de variabilidades e quanto tempo duram as discussões; e finalmente, qual é o perfil dos usuários do Stack Overflow que perguntaram e responderam as questões recuperadas, bem como o grau de confiança que se pode ter sobre a qualidade das perguntas e respostas apresentadas. Um dos principais resultados desta pesquisa foi a definição de uma lista de palavras que se referem a conceitos e técnicas de implementação de variabilidade que são mais comumente usados pela comunidade de desenvolvedores do Stack Overflow. Outro são os resultados da pesquisa a partir da análise das questões e respostas selecionadas, tais como: dois grupos de palavras foram identificados como os mais utilizados pela comunidade Stack Overflow nas perguntas: um mais conceitual e um mais relacionado às técnicas de implementação de variabilidade; o tempo médio de atraso para a primeira resposta às perguntas e a duração da discussão quando há mais de uma resposta; e finalmente, um estudo sobre o grau de confiança do grupo de usuários do site Stack Overflow que fez perguntas ou respondeu perguntas com base no indicador de reputação dos membros da comunidade.

Palavras-chave: Implementação de variabilidades, Reúso, Stack Overflow, Latent Dirichlet Allocation, Linha de Produto de Software

ABSTRACT

In this dissertation an empirical investigation was carried out aiming at understanding how the theme "implementation of variability" is discussed by the community of Information Technology professionals who use the Stack Overflow site. Initially, a list of 52 words related to this theme was defined based on the literature of the area and the opinion of experts. They were then used as the initial vocabulary for the algorithm that implements the Latent Dirichlet Allocation (LDA) in order to find the main topics related to variability implementation, as well as to validate the words in the list. Then, with the most used words in the questions and answers, as validated by LDA, searches were performed on the dataset retrieved from the Stack Overflow site and several studies were carried out to find the best set to be analyzed, considering that some words used alone recovered many questions, because they are other uses in the IT area. The best result was obtained by combining these words two by two, and thus 1962 questions were retrieved. These 1962 questions were then analyzed with the objective of discover the mechanisms of implementation of variabilities more discussed, less discussed and not discussed; what is the delay in responding to the formal questions on implementation of variabilities and how long the discussions last; and finally, what is the profile of Stack Overflow users who asked and answered the questions retrieved, as well as the degree of confidence one can have about the quality of the questions and answers presented. One of the main result of this research was the definition of a list of words that refer to variability implementation concepts and techniques that are most commonly used by the Stack Overflow developer community. Other one are the research results from analyzing the questions and answers selected, such as: two groups of words were identified as those most used by the Stack Overflow community: one more conceptual and one more related to variability implementation techniques; the average delay time for the first answer to the questions and the duration of the discussion when there is more than one answer; and finally an study on the degree of confidence in the Stack overflow user group that asked questions or answered questions based on the reputation indicator of community members.

Keywords: Variability implementation, Reuse, Stack Overflow, Software Product Line, Latent Dirichlet Allocation

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface	15
CC	Conditional Compilation	40
DOM	Document Object Model	15
FOP	Feature Oriented Program	25
GWT	Google Web Toolkit	14
LDA	Latent Dirichlet Allocation	9
LSI	Latent Semantic Indexing	9
MVC	Model, View, Controller	40
pLSI	Probabilistic Latent Semantic Indexing	9
SPL	Software Product Line	17
TAN	Technology Associative Association	15
TE	Topic Entropy	29
XML	Extensible Markup Language	19

Chapter 1

INTRODUCTION

1.1 CONTEXT AND MOTIVATION

Satck Overflow is an online platform where users can share knowledge by asking or answering questions. The area of software engineering – as well as other areas of computer science – currently uses this type of information source to solve doubts and share knowledge. It is also a kind of social network that brings together people with a common interest.

The Stack overflow dataset has relevant information in the area of software engineering since the year 2008, when it began its operation. Analyzing the textual content of this data set can help to better understand the thoughts, doubts, and needs of software developers. Questions asked by practitioners can be understood as the difficulties they face in their work.

There are several studies published using the Stack Overflow content, with several questions that have been asked and answered in the field of software engineering related to different themes, such as: mobile software development, program design and programming technologies (TREUDE; BARZILAY; STOREY, 2011; MAMYKINA et al., 2011; BARUA; THOMAS; HASSAN, 2014; VENKATESH et al., 2016).

However, studies on the implementation of variability in the context of Software Product Lines have not yet been done and published. The major research papers related to the study of variability implementation and management usually focus on systematic mappings or empirical research that uses in their studies a relatively small number of participants when compared to the large active community of Stack Overflow (BABAR; CHEN; SHULL, 2010; CHEN; BABAR, 2011; VILLELA et al., 2014).

In order to have a solid ground to base our master's dissertation, we did not find any study in the area with the amount of information that can be found in the Stack Overflow dataset, since the platform has a total of 7.3 million of registered users, who can ask and answer questions, comment and vote.

On the other hand, analyzing the textual content of such a large repository of knowledge poses a number of challenges. One is that the total volume of data prohibits manual analysis. In addition, the unstructured nature of formulated questions and answers that are written in natural language and mixed with pieces of code render inefficient the most conventional data mining techniques (BARUA; THOMAS; HASSAN, 2014).

1.2 PROBLEM STATEMENT

The literature does not present studies related on the topic of implementation of variability as well as in the more general context of software reuse and software product lines, focusing on the practices of developers using the Stack Overflow dataset as a knowledge repository. By investigating the Stack overflow dataset, we can investigate topics related to the implementation of variability and thereby identify which difficulties the developers have reported.

1.3 OBJECTIVES AND CONTRIBUTIONS

The main objective of this dissertation is to analyze the database of questions and answers of the Stack Overflow site to identify what developers and users have commented, questioned and answered about implementation of variability in the contexts of software reuse. This will be done based on a set of seven research questions

The main contribution of this dissertation is the analysis of the data retrieved to answer each research question and the report of its findings.These questions investigate the following themes: which are the topics, keywords, mechanisms the most and the least used and discussed in the questions related to implementation of variabilities? How fast this occurs and how reliable is this information?

One of the main result of this research was the definition of a list of words that refer to variability implementation concepts and techniques that are most commonly used by the Stack Overflow developer community in their questions. Besides that, we analyzed the set of answers retrieve based on five Research Questions. The main results of this analysis are summarized next. Two groups of words were identified as those most used by the Stack Overflow community: one more conceptual and one more related to variability implementation techniques; the average delay time for the first answer to the questions and the duration of the discussion when there is more than one answer; and finally an study on the degree of confidence in the Stack overflow user group that asked questions or answered questions based on the reputation indicator of community members.

1.4 RESEARCH THEMES OUT OF THE SCOPE OF THIS DISSERTATION

The following topics are not considered in the scope of this dissertation:

- Studies aimed at evaluating the best topics recovery tools. In our literature review, we have identified the LDA algorithm as the most used as a tool for retrieving topics, so we will not perform any type of study to evaluate topic recovery algorithms.

- Studies focused on mobile devices and their operating systems. In our literature review, we found studies related to Stack Overflow that had as main objective to analyze the development of mobile devices to take into account variability management.

Questions or answers that have been removed or closed by Stack overflow moderators will not be considered in the survey.

1.5 DISSERTATION ORGANIZATION

This dissertation is organized as follows. Chapter 2 presents background information to the the research problem main approaches. They describe succinctly the main functionalities of the site Stack Overflow and the main concepts of software variability implementation. Chapter 3 presents related work on researches with Stack Overflow, reuse and variability implementation. Chapter 4 describes the research methodologies developed and used in this research. Chapter 5 presents the experimental results obtained. This is done through seven Research questions that are posed and answered. At the end, the results are summarized end discussed. Finally, in Chapter 6, a summary of this dissertation is presented as well as its main contributions and ideas for future work.

Chapter 2

BACKGROUND INFORMATION – STACK OVERFLOW AND VARIABILITIES

2.1 INTRODUCTION

In this chapter some of the main concepts and subjects involved in this dissertation that are important for the understanding of the research carried out are defined and discussed. Thus, as the main target of this work, the Stack overflow site is approached in the first three sections: in Section 2.1 is presented an overview of the site, in Section 2.3 are summarized the main functionalities made available to its users and in Section 2.4 the concept of trust indicators that make up the profile of users is defined and it is detailed how the Stack Overflow website has implemented such an indicator, namely Reputation. Section 2.4 presents the Latent Dirichlet Allocation algorithm, which aims to find sets of words that form topics in texts. Finally, section 2.6 defines and discusses the concept of implementation of variabilities, which is the main theme of interest of this dissertation.

2.2 A GENERAL OVERVIEW OF STACK OVERFLOW

Stack overflow is an online platform in which users exchange knowledge through questions and answers to doubts about topics related to programming and software engineering tasks. The platform allows users to ask new questions and answer existing questions, as well as evaluate the questions and answers based on the perceived value of the post. All user-generated content is licensed under a Creative Commons Attribute-Share-alike license

The use of online platforms such as Stack Overflow has changed the way programmers clarify their doubts. On the other hand, the large accumulation of data in this database has created the opportunity for researchers to analyze them in a variety of ways. It is possible to investigate the search for knowledge of the developers and the factors that influence their various questions and doubts using a sample mining methodology, which is a basic search performed on the site search engine mechanism (THOMAS et al., 2011).

Often, software programmers and hobby programmers look for answers to questions using sites like Stack Overflow. The analysis of these data has potential to provide information on various aspects in the field of software development, such as technology trends, most commonly used and most cited programming languages, development topics that most generate doubts (e.g. testing, human-computer interface, search algorithms) (TREUDE; BARZILAY; STOREY, 2011).

Users who visit Stack Overflow without creating an account are invisible. Interestingly, many of these individuals can find answers to their questions through the search engines, making sure whether other developers have asked a similar question. The site receives on average seven million visitor monthly (MAMYKINA et al., 2011).

Figure 2.1 shows the home page of Stack Overflow with the platform number of registered users. There are registered users in Stack overflow, a group with about a quarter of the total's users, who have not yet asked, answered or voted. The second most frequent group consists of users who only ask but do not answer questions (23.7%), followed by users who only answer but never ask or vote (20.4 percent). Overall, almost half (48.5%) of the registered users have already answered a question. The overlap of users who ask and answer is significantly greater: 16.4 + 7.7 = 21.4% (MAMYKINA et al., 2011).

In the first two years of its creation in 2008, more than one million questions were asked and more than 2.5 million answers were offered (TREUDE; BARZILAY; STOREY, 2011). More than 92% of the questions about specialized topics are answered in an average time of eleven minutes (MAMYKINA et al., 2011). The data extracted in 2016 contained millions of posts created by ten thousand developers, totaling about 21.7 million messages and 32.5 million comments (ZOU et al., 2017). Figure 2.2 displays the total of non-Stack Overflow queries until July 2017, showing a total of more than fourteen million questions.

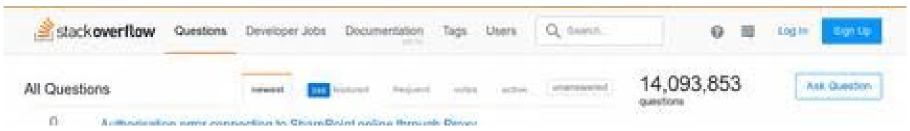


Figure 2.1 StackOverflow: Total of Questions - July 2017

2.3 MAIN FUNCTIONALITIES OF STACK OVERFLOW

In order to achieve the objectives of this research and to answer the research questions formulated, it is important to understand what the role of Stack Overflow users is, what are the main actions they can perform on the site, i.e. what are the functionalities of the site and how the interaction between users occurs.

User actions are commands that can be executed in a question or in its constituent elements. The actions available to users are:

- **Question**: When users write a new question.
- **Answer**: When users provide answers to a formulated question.

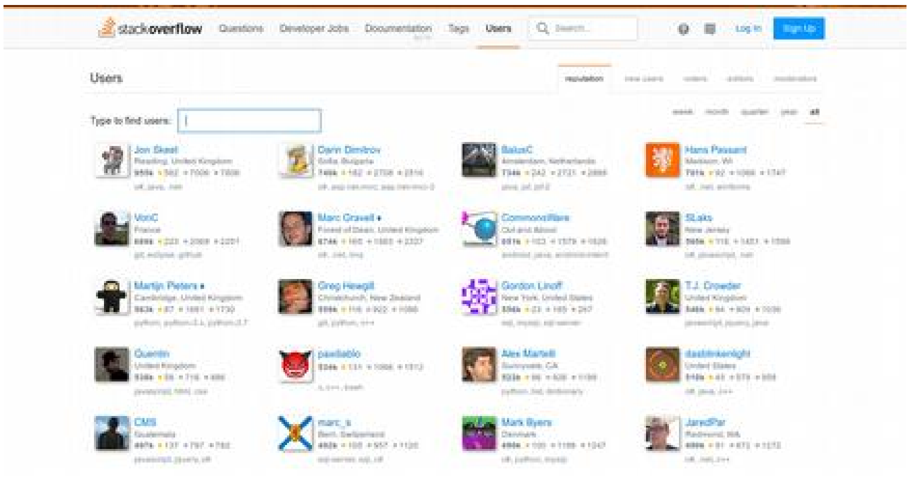


Figure 2.2 StackOverflow Home Page

- **Accept Answer**: When the user who asked the question chooses an answer as having answered a question correctly.
- **Choose favorite answer**: When users with good reputation indicate a question or answer as their favorite.
- **Vote**: When a user with a good reputation indicates (or "votes") positively or negatively a question or answer.
- **Retrieve**: Any visitor – public or registered – can retrieve and view questions and answers using keywords and tags in the search.

When a question is, for example, poorly written, out of site goals or very generic, it can be removed or closed by moderators. The platform's own community is responsible for its organization and functioning, defining the role that more experienced users can play, such as the role of moderator. Users receive new privileges with an increase in reputation, like the right to vote, comment, and even edit other people's posts. The evaluation of answers is made by "votes" and the importance of questions and answers is given by counting the number of votes, number of visits and number of answers they obtain (ZOU et al., 2017).

It is important to notice that this process provides a content curation for the site that contributes to its quality. A detailed description of the rules for reputation copied from the Stack Overflow site is presented in Appendix A.

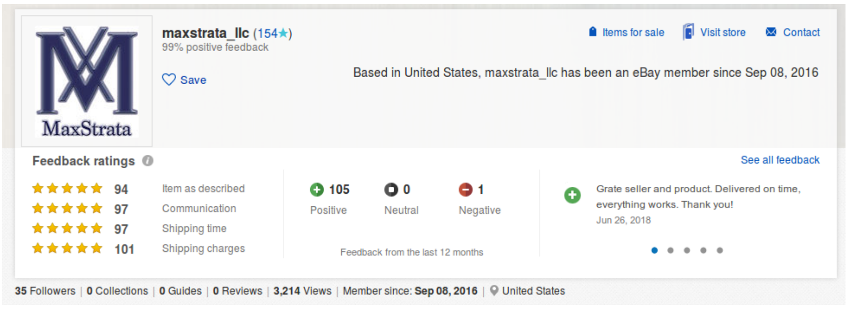
2.4 REPUTATION AND TRUST

In online systems, reputation is a result of the assessments that users of a community realize about the use of some product, service or even another user. Based on these evaluations a reputation value can be obtained for the user or object. Resnick et al. (2000) discuss the pros and cons of reputation systems from some online sites, such as eBay, expert systems sites such as www.askme.com, and iExchange.com. They conclude that they appear to perform reasonably well and there's no better way to earn one another's trust in online interactions.

Trust is a word used in many areas and each has its own specific definition. For psychology, trusting a person is a commitment to make, an action based on the belief that that person's future actions will lead to a successful outcome (GOLBECK, 2005). Typically, an individual with a good reputation tends to influence others and this induces the people with whom he or she interacts to have high trust. There is no specific rule for relating trust to reputation. In certain situations particular interests can influence these concepts and the trust is then defined as a personal and subjective factor.

One can consider that trust in an individual is derived from a combination of references received and personal experience. On the other hand, reputation is based on lack of personal experience, it is a measure of collective trust, calculated on the basis of the references or qualifications of the members of a community, ie trust is based on third-party references (GOLBECK, 2005).

This type of system helps people decide who to trust by encouraging interaction behavior. In their work, Jøsang e Golbeck (2009) state that the purpose of systems of trust and reputation is to strengthen quality in communities by providing incentives for good behavior and punishing otherwise. Reputation systems are used in successful online business applications, where trust that generates a reputable user is important


maxstrata_llc (154★)
99% positive feedback

Save

Items for sale | Visit store | Contact

Based in United States, maxstrata_llc has been an eBay member since Sep 08, 2016

Feedback ratings — See all feedback

★★★★★	94	Item as described
★★★★★	97	Communication
★★★★★	97	Shipping time
★★★★★	101	Shipping charges

Positive	Neutral	Negative
105	0	1

Feedback from the last 12 months

Grate seller and product. Delivered on time, everything works. Thank you!
Jun 26, 2018

35 Followers | 0 Collections | 0 Guides | 0 Reviews | 3,214 Views | Member since: Sep 08, 2016 | United States

Figure 2.3 Ebay: Reputation of a seller

Golbeck (2005) considers that social networks are developed with common Internet users in mind and so they are simple tools. With this, the definition of trust should be as simple as possible so that users can understand what they are expressing and do it accurately. An example of how users are rated and control their reputation on the Ebay websit[1] is shown in Figure 2.3.

This work focuses on Stack Overflow and its collaborative social interactions, and in that context reputation represents the group's opinion about someone. It is, therefore, an indicator of confidence, and it can be concluded that the reputation system of the Stack Overflow website meets the definitions and requirements listed by the authors cited in this section.

[1]www.ebay.com

A user's reputation is built over time based on their participation using the site. It is translated into a numerical value that is calculated by a formula that takes into account the number of questions and answers of the user and the votes that their answers receive, as well as other aspects of their interaction. This value is a positive integer and the reputation is higher the higher this number. Thus, this value starts with the number 1 and goes up indefinitely. There are users with reputation higher than 100.000.

Users of Stack Overflow can also be symbolically "paid". A person, for example, is awarded 10 reputation points for receiving an "up" vote on an answer given to a question and 5 points for the "up" vote of a question, and can receive badges for their valued contributions, like gold, silver and bronze medals.

2.5 LATENT DIRICHLET ALLOCATION (Latent Dirichlet Allocation (LDA))

Currently, the information generated has increased exponentially, making it increasingly difficult to find or interpret information stored in documents. With the intention of executing this task, probabilistic techniques called topic modeling have been developed. They are used to discover, extract and group documents from large collections in thematic structures (HOFFMAN et al., 2013).

One of the most representative algorithms used in topic modeling is the Latent Dirichlet Allocation (LDA). In modeling, each document is represented as a combination of topics and each topic is represented by a set of terms, both with associated probabilities. Thus, each topic extracted from the collection has more relevant terms (BLEI; NG; JORDAN, 2003). These algorithms are widely used in the organization of textual documents (HOFFMAN; BLEI; BACH, 2010). Currently, its use expands to several scenarios, such as academic and technical documents, news and social networks (HONG; DAVISON, 2010).

A significative step forward in this regard was given by Hoffman, Blei e Bach (2010), which presented the probabilistic model Latent Semantic Indexing (LSI) (pLSI), also known as the aspect model, as an alternative to LSI . In the Probabilistic Latent Semantic Indexing (pLSI) approach, the models of each word are placed in a document as a sample of a combined model, where the components of the combination are multinomial random variables that can be viewed as representations of "themes". Thus, each word is generated from a single topic, and different words in a document can be generated from different topics. Each document is represented as a mixed list of proportions for these components of the mixture and thus reduced to a probability distribution of a fixed set of topics. This distribution is the "reduced description" associated with the document.

Regarding probabilistic topic models, the most well-known and cited model in the area literature is LDA, which is a generative probabilistic model for discrete data collections, such as document corpus (BLEI; NG; JORDAN, 2003). Its basic idea is that documents are represented as random combinations on latent themes, where each topic is characterized by the distribution over words.

A generative model is one that randomly generates the data from the latent variables. Thus, LDA is not an algorithm with sequential descriptions of instructions for finding topics from a collection of documents. LDA is a probabilistic model that describes how

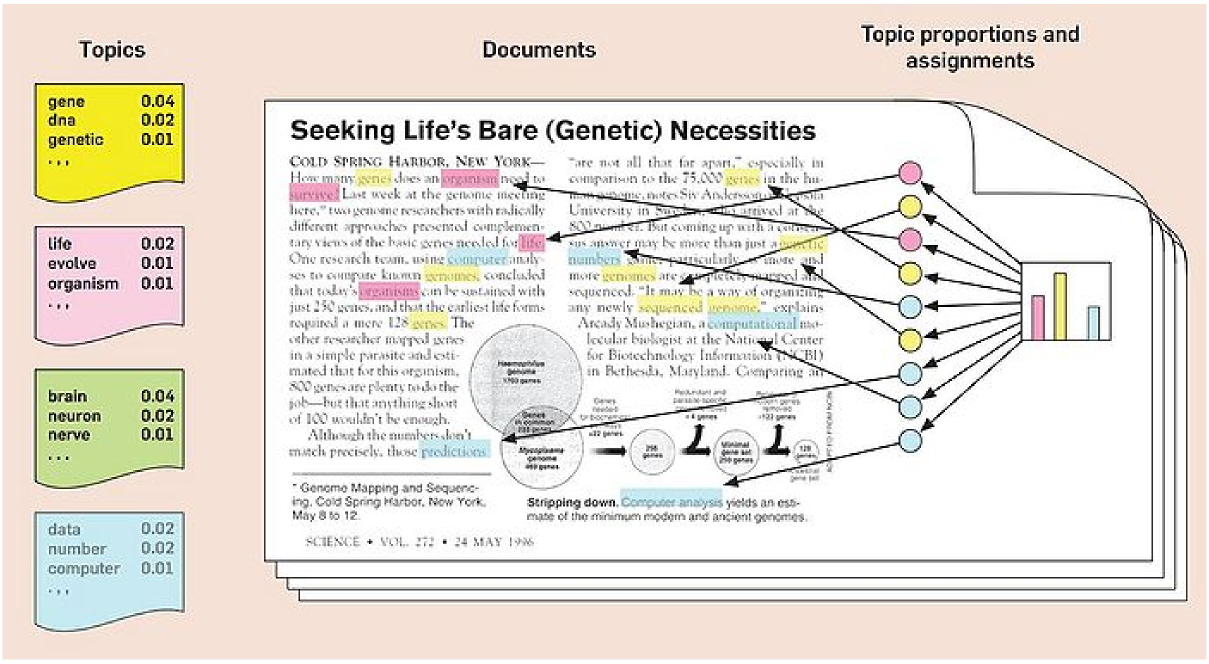


Figure 2.4 An overview of LDA

documents are generated. Figure 2.4 shows schematically this operation.

With the increasing amount of information generated, it becomes increasingly difficult to retrieve and interpret stored documents. For this task, probabilistic techniques called topic modeling have been developed, which are used to discover, extract and group documents from large collections in thematic structures. Online LDA is a good choice for use in large data sets because it needs to maintain a very small subset of the data set in memory at a given time and a good fit for data transmission since new batches of data can be fed continuously as are received (HOFFMAN; BLEI; BACH, 2010). In the online variation, only one scan of the entire data set is analyzed, analyzing one piece of document at a time. A “piece” can be a single document, multiple documents or even the entire data set.

Software engineering uses the source code of programming languages as a basic element and one of its characteristics is to be much more repetitive and predictable than natural language texts. Thinking about this, Panichella et al. (2013) warned researchers about the dangers of ad-hoc calibration of LDA in a software corpus, as was done predominantly in the software engineering research community, or using the same settings and parameters applicable only to natural language texts.

2.6 VARIABILITY IMPLEMENTATION

Software reuse is the process of creating software from existing software instead of developing from scratch. One technique that is directly related is the ability to organize and select what will be reused. Reuse of artifacts in software engineering is important to achieve the required productivity and quality improvements, such as: speed of delivery and cost reduction. Companies are shifting their business vision and software reuse is starting to appear in their schedules as a systematic and managerial process, focusing on system families, based on repetitive and controlled processes, concerned with large-scale reuse (BABAR; CHEN; SHULL, 2010).

A software product line (SPL) is a type of system that incorporates the main common parts of a domain and reuses them in different products (or specific systems). Each product, in turn, may have specific parts valid only for it and for some other products, but not for everyone. It is then said that a SPL contains variations, which characterizes it as different from other specific software. Variations generally refer to a feature (or functionality) that the SPL implements (CLEMENTS; BACHMANN, 2005).

Variability refers to the ability of a software to be configured, customized and changed to generate different products depending on its context. Variability management encompasses the activities of: representing software variability, managing dependencies between different options, and supporting the instances of these variabilities. These are extremely complex and challenging tasks that need to be supported by appropriate approaches, techniques and tools.

All variability found in a software product line can be connected to a corresponding feature whose support varies under specific conditions. The implementation of variability is almost always spread across multiple files or software modules. When there are interdependencies between the implementation modules, they must be explicitly clarified. Otherwise, maintenance of the modules can cause serious defects in the software product line (GACEK; ANASTASOPOULES, 2001).

To avoid variation problems, when several points of variation between resources become complex, and their interactions are a risk for the proper functioning of a new product, variability management should be employed. This is primarily intended to solve the problems of a large number of variations and their complex dependencies, and therefore relying on this type of evidence can be quite challenging for transition and use in projects.

Additional assistance in managing code-level variabilities can be made as long as the exact variability linkage time is known. Gacek e Anastasopoules (2001) classified the bonding time as follows:

- Compile time: variability is resolved before compilation of the actual program (for example, with pre-processor directives) or compile time.
- Link-time: variability is resolved during module or library link (for example, selecting different libraries with different versions of the exported operations).
- Run-time: variability is resolved during program execution (for example, depending on user rights features are disabled or enabled with conditions in the code)
- Update time or post-run time: variability is resolved during program updates or after program execution (for example, an update utility adds functionality to existing modules)

This recommendation applies to the following affected techniques:

- Aggregation/Delegation,
- Inheritance,
- Parameterization,
- Overloading,
- Dynamic Class Loading,
- Static Libraries,
- Dynamic Link Libraries,
- Conditional Compilation,
- Frames,
- Reflection,
- Aspect-oriented programming,
- Design Patterns

After having the resource type and connection times identified, it is also necessary to determine the variation parameters. The main variation parameters identified are the interfaces and corresponding implementations (GACEK; ANASTASOPOULES, 2001).

In order to ensure effective maintenance and systematic change, management traceability must be provided from the architecture and design to the code. Traceability is the ability to document and follow the life of a concept throughout the development of the system. It can occur in two ways: directed forward (post-traceability: describing the implementation and use of a concept) as well as directed backwards (pre-traceability: describing the origin and evolution of a concept).

A standard way to provide traceability is to set the date and reference. Such references can be expressed as links or arrays in which the connections between the various artifacts in code and the architecture/design are made explicit. This assumes that the artifacts to be traced were identified for the first time, which brings again the essential need to determine the parameters and the points of variation.

An important issue faced in this type of project is the scalability problem. Many of the implementation approaches used in industry to manipulate variability in a product line suffers when a new product is entered or when existing products are evolving. This first impacts the overview of the points of variation that gradually lead to a degraded implementation structure. In many cases developers try to manage the next growth and evolution by revising their implementation approaches or introducing step-by-step new procedures.

Chapter

3

RELATED WORK: STACK OVERFLOW, LDA AND VARIABILITIES

3.1 INTRODUCTION

We started our work by searching online libraries such as ACM, IEEE and Springer, looking for articles related to the Stack Overflow platform and for techniques of variability implementation in the period between 2000 and 2017. After selecting the relevant articles, we have looked in the references of these publications to add more articles relevant to our research. Among the studies of Stack Overflow there are the ones that mine data, i.e. questions and answers, and others that analyze it generically.

Other researches aim at evaluating how variability impacts the development of software in the context o software product line. Developers of this type o system are the target of our research as they are the ones who may ask question on Stack Overflow. Although there many papers published regarding implementation of variabilities in the context of software reuse, we have found none specifically linked with Stack Overfow.

For the organization of this chapter, the published articles have been separated into three groups: those that performed general studies about Stack Overflow (Section 3.2); those that performed studies mining questions and topics (Section 3.3); and articles that defined and studied software variabilities (Section 3.4). Conclusions are reported in Section 3.5

3.2 CHARACTERISTICS OF THE STACK OVERFLOW

Over time, platforms such as Stack overflow have become repositories of knowledge of software engineering. Such repositories of knowledge can be invaluable information on the use of specific technologies and the tendencies of the discussions (BARUA; THOMAS; HASSAN, 2014). This resulted in published scientific papers aiming to analyze the content of Stack overflow to categorize their questions (TREUDE; BARZILAY; STOREY, 2011) and to identify their characteristics (MAMYKINA et al., 2011).

During the development of this work, several data mining technologies used in software engineering were studied.These technologies have objectives similar to the study presented in this dissertation. There are some articles that analyzed Stack Overflow in this line of research. Chen e Xing (2016) present an overview of information mining in Stack Overflow and propose a graphical tool to represent this scenario. Calefato et al. (2015) found evidence after mining Stack Overflow that information, time and affection may increase the chances of a user have his(her) answer accepted. Another mining work in Stack overflow was Bajaj, Pattabiraman e Mesbah (2014), which did a mining to sort all the questions based on their importance. Each one of these works presented its own mining approach and this scenario encourages us to also present our approach synthesizing parts of these works.

Some researchers have been interested in Stack Overflow to understand, for example, how it works, how it is organized and why it succeeds. One such study is that of Mamykina et al. (2011), who analyzed the patterns of user interaction with the platform. This helped to highlight some of the most common user groups and their behaviors. In their qualitative study they discussed the lessons learned from the Stack Overflow study, explained the users' behavior and presented the reasons for the site success, such as:

- founders having involvement with the community,
- high responsiveness, with an interactive project approach, and
- an incentive system that promotes the user's desirable behavior.

They also studied the Stack overflow site to find topics and classify the issues into distinct types such as "as-a", "discrepancy", "environment", and so on. The authors applied their analysis using a period of 15 days and manually codified the questions based on a sample of the data. The main design features that led to the popularity of Stack overflow were identified: the point-based reputation system, the strong community involvement and the main focus on development (THOMAS et al., 2011).

Thomas et al. (2011) investigated the impact of technologies using modeling of topics. They identified and compared the impact of the technology in different domains. Stack Overflow benefits this approach by using a hierarchical method as a markup scheme, where the context of a question is used to place it in a broader category. Subsequently, the tags specified by the authors are used to improve categorization.

According to Treude, Barzilay e Storey (2011), a possible reason for the high degree of answers to the issues (question) are the fact that the issues are usually very concrete. They contain snippets of code and often the fonts are not enough to understand the code and make recommendations about its quality. In addition, questions can have more than one "correct" answer and often any answer is better than none.

In addition, Parnin, Treude e Grammel (2012) conducted an empirical study to investigate how Stack Overflow can facilitate the documentation of popular APIs such as: Android, Google Web Toolkit (GWT) and the Java programming language. They have identified that the Stack Overflow users are able to generate a rich source of content with code and discussion examples that is actively viewed and used by many developers. However, they found deficiencies in the documentation .

Chen e Xing (2016) presented a technology scenario extracted from Stack overflow and represented it in a graphical tool called Technology Associative Association (TAN). In their empirical study they showed that Technology Associative Association (TAN) captures a wide range of technologies and shows the relationships between technologies and their trends.

In their work, Zou et al. (2017) used the Stack Overflow dataset to understand which evolutionary tendencies and difficulties were presented in relation to non-functional requirements. They found that developers discuss more about usability and reliability compared to maintenance and efficiency. Regarding the evolution of the discussions, the functionality and reliability of the non-functional requirements attracted the attention of the developers.

Once we identified the need to understand how the LDA algorithm works, we added to our review the articles related to LDA found as reference in the studies. With this we come to understand that there are some variations of the algorithm and that to meet our study the most appropriate variation is the online, developed by Hoffman and colleagues (HOFFMAN; BLEI; BACH, 2010).

3.3 STACK OVERFLOW, QUESTIONS AND TOPICS

An example of article that uses LDA is the one authored by Kavaler et al. (2013), which aimed to understand what information programmers are looking for, and why. For this they have joined the data from Android Market and Stack overflow. They studied the relationship between the use of Android Application Programming Interface (API) classes in over a hundred thousand applications and the questions and answers that mention these classes, so they concluded that:

- questions increase with use, although there is a non-linear saturation effect, suggesting that the knowledge of the most popular classes is internalized to some extent or becomes easier to find;

- the total amount of documentation available for a class has no impact on the number of questions but to the size of the classes, suggesting that more complex classes increase the need for information;

- the situation is reversed for answers: the total amount of class documentation decreases the number of answers per question, while the class size has no effect and the documentation by method increases the number of questions.

Bajaj, Pattabiraman e Mesbah (2014), on the other hand, conducted an empirical study of web development discussions in the Stack Overflow dataset, to understand common difficulties and misconceptions among developers. This study involved a text analysis of questions and answers related to web development to extract the dominant topics of discussion using topic models based on the LDA algorithm.

The results show that browser-related discussions, although predominant in the past, are becoming less important, that the Document Object Model (Document Object Model

(DOM)) APIs and forms handling problems have presented a significant source of confusion for the development of web, and that HTML5 is gaining popularity in (mobile) web applications.

Barua, Thomas e Hassan (2014) presented as a methodology to analyze the textual content of the discussions in the dataset of Stack overflow, a statistical modeling technique based on the algorithm LDA. Its objective is automatically discover the main topics present in the developers' discussions. They have also analyzed the topics discovered, as well as their relationships and trends over time.

In their analysis, they made a number of interesting observations, such as: (1) topics of interest to developers vary widely from works on version control systems for *C#* syntax; (2) questions on some topics lead to discussions on other topics. The topics that have gained more popularity over time are web development (especially jQuery), mobile applications (especially Android), Git and MySQL.

3.4 RESEARCHES RELATED TO VARIABILITY

Gacek e Anastasopoules (2001) addressed in their work how to deal with variability at the code level. They examined several implementation approaches in relation to their use in a product-line context. They concluded that different approaches were necessary to support different problems and that idioms as well as the programming language used in the implementation can play a significant role in the implementation process and should be taken into account (GACEK; ANASTASOPOULES, 2001).

In their work, Svahnberg, Gurp e Bosch (2005) described the relevant factors to determine how to implement variability and proposed a taxonomy of variability implementation techniques. They concluded that the implementation of variability needs to be managed during the product life cycle and that this restricts the choices of how to implement them. They also presented a minimal set of steps to introduce variability into a software product line.

Babar, Chen e Shull (2010) conducted a study based on a systematic review to synthesize and evaluate evidence on the efficacy of solutions proposed for management of variability. The study evaluated 97 articles selected over 20 years of research, providing an overview of the temporal distribution of the types of evidence reported in the studies.

In addition, Chen e Babar (2011) conducted an empirical study using focus group as a data collection method to identify the challenges faced by professionals regarding variability management. The technical issues identified were: complexity manipulation, knowledge collection and management, extraction of technical artifact variability, variability evolution, decision management and execution and testing.

Villela et al. (2014) analyzed the answers of 31 respondents from thirteen countries and discovered the existence of a correlation between the business domain and the variability management approaches used. They also found that the difficulty of ensuring the quality of maintenance due to the explosion of dependencies was pointed out as a relevant problem. The main findings highlighted by the study were:

- there are some predominant variability management approaches in specific phases, but most phases present a heterogeneity of approaches,

- the commercial domain is the main factor that influences the variability management approaches adopted.
- despite the growing maturity of the research field, there is evidence that experts still face difficulties in systems engineering when they are rich in variants,
- the main difficulties faced by professionals are related to the evolution of systems rich in variants,
- the type, purpose, complexity of variant-rich systems and, especially, the business domain, influence the perception of difficulties in using, developing, or evolving systems.

Bosch, Capilla e Hilliard (2015) state that software variability concerns all lifecycle phases, from requirements elicitation to post-deployment and execution time. In addition, selection of a variant can affect the rest of the system because of dependencies between variation points and variants. They concluded that context resources should be identified through context variability analysis and captured in resource models.

Vale et al. (2017) analyzed 62 studies from a period between 2001 to 2015 and discussed seven aspects of software product line traceability: strategies, application domains, research intensity, research challenges, accuracy and industrial relevance. They provided a structured understanding of Software Product Line (SPL) traceability and concluded that it is in the process of maturation.

To improve our understanding, we considered studies related to the implementation of variabilities, we also observed that many recovered studies use as a tool for the research the systematic review of the literature or survey. In such a work, Chen e Xing (2016) analyzed the challenges related to variability and identified that there was no systematic effort to study how the variability management approaches reported were evaluated. They did a systematic review of the literature on variability management approaches in software product lines, reported between 1990 and December 2007. They found that only a small number of the approaches reviewed were evaluated using rigorous scientific methods. A detailed investigation revealed significant quality deficiencies in several aspects of the quality assessment criteria used.

3.5 CONCLUSION

The main difference from the work reported in this master dissertation and the previous works is the fact that our goal is to understand how the Stack Overflow community treats the implementation of variability in software development and what are the main problems and doubts the practitioners have when they implement variabilities in their software systems.

In the studies published using the Stack overflow dataset, we were able to understand what the academy has been studying, and we have identified that modeling of topics using the LDA algorithm is the most used form of research for the analysis of data in this context.

This bibliographic review also contributed to the definition of the research questions of this work and the queries made in the Stack Overflow database to define the relevant questions to be retrieved. It contributed as well to collect words and terms that are used by researchers related to variability implementation.

Chapter 4

RESEARCH METHODOLOGY

4.1 INTRODUCTION

Stack Overflow moderates hundreds of thousands of posts per month for developers with a wide variety of backgrounds, asking questions on various topics. Taken together, these posts act as a record of the history of the needs and thoughts of the developers (BARUA; THOMAS; HASSAN, 2014).

This work followed three major steps. The first was to understand the workings of the Stack Overflow community, understand its search engine, and how the community creates, answers, and votes on questions and answers. The second step was to recover the entire Stack Overflow records file and prepare it for offline use. The second step was to retrieve the data of interest for this study offline. This has been done in two distinct ways, which are reported in this chapter. The third step consisted of analyzing the data and reporting the results, which is reported in the next chapter.

It is important to inform that three researchers in the area of software engineering with experience in the field of software reuse have collaborated with the research reported in this work. Their participation took place with the purpose of validating decisions taken by the author as well as the results found. An example of this participation occurred in the definition of keywords chosen for the searches covered in this chapter.

This chapter is organized as follows. Section 2 describes how the original data from the Stack Overflow system was collected and organized in an off-line dataset. Section 3 describes how the dataset was prepared so that it could be used by the LDA algorithm searching for topics. Section 4 describes another way to prepare the data to be used in a keyword-based retrieval. Finally, Section 5 concludes the chapter with the list of research questions that will be used in the next chapter to analyze the data collected and prepared.

4.2 CREATION OF AN OFFLINE DATASET

Stack overflow data is publicly available in Extensible Markup Language (XML) format under the creative commons license. The data set is divided into five XML documents:

badges.xml, comments.xml, posts.xml, users.xml, and votes.xml. The posts.xml file has the actual text content of the posts, as well as the count of views, count of favorites, type of publication, date created, and user ID that created each post (TREUDE; BARZILAY; STOREY, 2011).

Retrieval of the dataset was done in April 2017 and resulted in a data set with 34,772,306 lines, divided into: 13,472,784 questions and 21,299,522 answers. Data from September 2008 to March 2017 were stored in the dataset, which is named Posts.xml. It can be retrieved at: https://archive.org/details/stackexchange.

Such a large volume of data prohibits manual analysis. In addition, the dataset is unstructured, the posts are written in natural language and may contain code, which prohibits conventional data mining techniques from being effective, as we can observe in the example shown in appendix A.

Lines (or records) in the Posts.xml file are formatted as rows and each row represents a question or an answer. It is a tagged data string defining the name of the data type, followed by the content. Questions have the sequence of fields, or record structure, shown in Table 4.1 and the answers have the structure shown in Table 4.2.

Table 4.1 Fields for the lines of a Question

Fields
Row Id
PostTypeId
AcceptedAnswerId
CreationDate
Score
ViewCount
Body
OwnerUserId
LastEditorUserId
LastEditorDisplayName
LastActivityDate
Title
Tag
AnswerCount
CommentCount
FavoriteCount.

In the row shown in the appendix A, for example, the record identifier (RowId) is 36, the record type (PostTypeId) equals 1 indicates that it is a question, vizualization counter (ViewCount) shows that that question was viewed 43086 times and the answer counter (AnswerCount) informs that it had 8 answers.

It may be noted that some items are common to both types of records and others are specific, such as AcceptedAnswerId for questions and ParentId for answers. These values

Table 4.2 Fields for the lines of an answer

Fields
Row Id
PostTypeId
ParentId
CreationDate
Score
Body
OwnerUserId
LastEditorUserId
LastEditorDisplayName
LastEditDate
LastActivityDate
CommentCount
CommunityOwnedDate

are used to interconnect questions and answers. In this case, for example, AcepptedAnswerId indicates that the answer accepted as correct is at line 352.

Some fields were added over time, such as Tag, which in the 2008 and 2009 question lines did not exist. These differences and the lack of a format pattern for lines of the Posts.xml file make it difficult to mine data of Stack Overflow.

The methodology used in this study was divided into two distinct parts, one based on topic modeling using the LDA algorithm to identify the topics and the other making direct query to the dataset. These methodologies are described in the following two subsections.

4.3 METHODOLOGY 1 – LATENT DIRICHLET ALLOCATION

This research methodology is based on topic modeling using the LDA algorithm, which has already been successfully used to retrieve textual content topics from several surveys (TREUDE; BARZILAY; STOREY, 2011). It will be used to identify as many of the possible terms related to the implementation of variability and to group them into topics.

For the purposes of this research, a list of terms was initially created with 132 terms, which are words retrieved from studies related to the implementation of variability. Then each term was used to search for its occurrence using the Stack overflow site search tool. After obtaining the search results, a worksheet containing the terms and the search result was created. This worksheet was validated by a set of experts to indicate which should and should not be removed. At the end of the process, there were 52 terms that were used as the basis for the LDA algorithm.

The next step was to analyze the content retrieved from the terms. At this point we identified that the term SPL (Software Product Line) retrieved values that did not represent the study of implementation of variability, since for the community of Stack

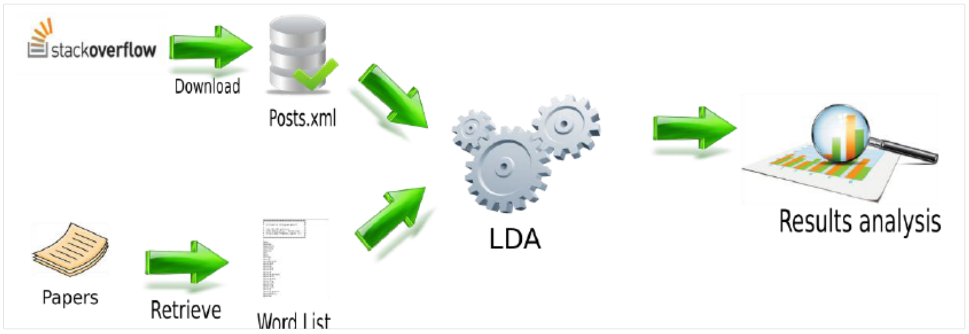


Figure 4.1 Methodology LDA

overflow it referred to "Standard PHP Library". This keyword was then changed to "Product Line".

Additionally, many of the texts retrieved using the keywords have been representing terms of object-oriented programming. In order to mitigate this failure we have created with the help of specialists a set of keywords, so that we can combine them with the terms and thus reduce the risk of recovering contents that are not related to the implementation of variability. These keywords can be seen in the first column of Table 4.5.

The online LDA algorithm was used in our research because it is a version indicated for studies with very large text corpora. A lot of time is spent to run the LDA algorithm directly in the Posts.xml file, so the original file has been separated into: questions, answers, questions with accepted answers and questions with no accepted answers. Table 4.3 displays the total number of questions and answers found according to this classification.

Table 4.3 Size of files after retrieval and separation

Classification	Total	Percentage	Percentage
Total of Questions	13.472.784	38,75%	
Questions with accepted answers	7.299.250		54,18%
Questions without accepted answers	6.173.534		45,82%
Answers	21.299.522	61,25%	
Total	34.772.306	100%	100%

Table 4.4 shows that, in average, in the site Stack Overflow, questions receive 1,58 answers. It shows as well that from theses answers, in average, a little bit more that half of the questions have an answer accepted as correct for the community.

Any user registered in Stack Overflow can vote on a question only once or, in other words, if an answer has 5,105 votes in total this means that this is the total number of users who have accessed the questions and voted on it positively. Usually users vote when a question (or even some answer) had value to him or her. To vote, it is necessary that the user has accumulated more than 300 points of participation in the site.

There is also the possibility for a user to indicate that a question is his or hers favorite. Favorite works as a "like" in social networks and it is an action reserved for those who

Table 4.4 Averages of Answers

	Average
Answers per Question (total)	1,58
Answers Accepted per Question	0,54
Votes	1,85
Indication as favorite	0,58

have less than 300 participation points on the site and also for unregistered visitors. It is therefore a more informal manifestation than the vote.

Table 4.4 also shows the average of votes and the average of indications for favorite for the questions retrieved in the dataset. Note that the average number of votes is much higher than the number of favorites.

Then, an algorithm was used to clear the textual content of the extracted messages and to discard any source fragments that were present in the messages. Keywords in XML, as well as common words, represent noises and do not help finding useful topics. We focus only on the contents of the following record components that store questions and answers in the Stack Overflow database: BODY, TITLE and TAG.

After generating the files, the LDA algorithm was used to retrieve the topics related to this study from the terms provided, and used as a basis to answer questions RQ1 and RQ2. The following decision was made: if the resulting set of terms has five keywords, this topic will be considered for the study, taking into account the number of 52 words used in the list and the number of 10 topics that will be retrieved.

4.4 METHODOLOGY 2 - DIRECT RETRIEVAL

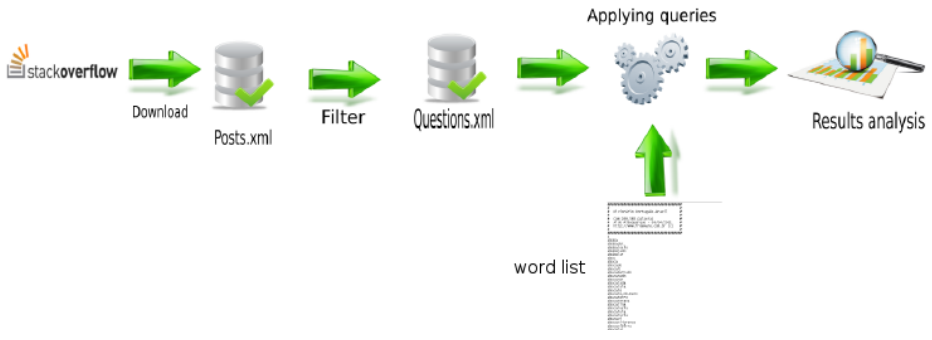


Figure 4.2 Methodology Direct Retrieval

As a second methodology, described in figure 4.2, the mapping technique was used using a set of keywords related to the implementation of variability. This set of keywords, shown in the second column of Table 4.5, was created with the support of the experts mentioned above. The tool provided by Stack Overflow was used to retrieve individual retrieval results from the search using these keywords as input. The results obtained are shown in Table 4.5, column two.

Table 4.5 Results produced by the words chosen

Index	Word	Total
1	# aggregation	38,087
2	# commonality	946
3	# conditional compilation	2,709
4	# decision model	18
5	# delegation	17,298
6	# design pattern	24,572
7	# feature dependencies	26
8	# feature interaction	7
9	# feature model	63
10	# ifdef	2,415
11	# ifndef	946
12	# parameterization	1,601
13	# product line	572
14	# product family	150
15	# product derivation	0
16	# reflection	74,115
17	# variability	1,381
18	# variant	45,745
19	# variation point	16

One can see that the results express the questions and answers found. For this work the main focus is on the problems related to the implementation of variabilities and, thus, the answers were discarded. For the desired combination, the term product derivation was removed as it did not obtain results individually.

In order to ensure that only the questions would be used, a subset of the Posts.xml file containing only questions was used. This file was named Questions.xml.

The keywords were combined two by two to get the files with the questions related to the combinations. The whole set o term combinations is presented in Appendix C. Then the files were put together in a spreadsheet to facilitate the analysis of the data. After that the results were reduced to 836 questions.

In order to improve the result a new approach with individual words was generated, from a new set of words proposed by the experts and presented in Table 4.6, with that we recovered a total of 2531 questions. Then, we inspected manually all these questions to remove the ones that were out of the context of our work and the set was reduced to 1962 questions.

The next step was to separate questions to search for other words that specialists said were also related to the implementation of variability, this was done to validate the research and it is presented in more detail in chapter 5, in the context of the preparation to answer the questions from RQ3 to RQ5.

Table 4.6 List of keywords

Keyword
Commonality
Variability
Decision Model
Feature Interaction
Feature Dependencies
Feature Model
Feature Oriented Program (FOP) (Feat. Orient. Prog.)
Product Line
Product family
Variation Point
Conditional Compilation
ifdef
ifndef

Then, a report file containing the fields ID, BODY and TAG of all the questions selected in the study was created for the experts to read and validate individually. After the questions were validated, a summary worksheet was created with the results that were used to answer the search questions from number 3 to 6.

4.5 CONCLUSION: RESEARCH QUESTIONS

Concluding this chapter, the data collected will be analyzed in according with the seven research questions that are listed next. The first two questions will use the data collect using methodology 1 and the other five questions will use the data collected according with methodology 2.

- RQ1: What are the hot-topics that describe the answered questions related to variability implementation mechanisms in Stack Overflow?

  This RQ aims to investigate the topics that represent technical concerns in the questions with accepted answers that the community of Stack Overflow software developers faces in the implementation of variability.

- RQ2: What are the hot-topics that describe the unanswered questions related to variability implementation mechanisms in Stack Overflow?

  This question considers unanswered questions because we are also interested in topics related to problems and techniques that do not have accepted answers. This indicates that Stack overflow developers face still unsolved difficulties in implementing variability.

- RQ3: Do mechanisms of variability exist that are discussed by the Stack overflow community?

In this question we will investigate if there are variability implementation mechanisms discussed by the Stack overflow community and what are they

- RQ4: What are the techniques related to variability implementation mechanisms most discussed in Stack overflow?

  With RQ4, we intend to identify whether variability implementation techniques are most found in the Stack Overflow developer community discussions.

- RQ5: What are the mechanisms of variability implementation not discussed by the Stack overflow community?

  With RQ5 we will define if the techniques of implementation of variability that are little discussed in Stack overflow represent those that in practice are also little used.

- RQ6: How fast is the crowd at covering widely variability implementation mechanisms?

  With this question we will analyze the speed with which the Stack overflow community answers the questions related to the implementation of variability.

- RQ7: Can we rely on the stack overflow crowd?

  With RQ7 we intend to answer which are the questions answered and which are not answered in the context of implementation of variability, as well as to understand what are the reasons for the questions not being answered.

Chapter

5

ANALYSIS OF THE RESEARCH QUESTIONS

5.1 INTRODUCTION

This chapter presents the analysis performed on the data contained in the dataset retrieved from the Stack Overflow site, as described in Chapter 4, and is carried out based on seven research questions.

This chapter is organized as follows. Section 5.2 contains the answers to RQ1 and RQ2, based on the analysis of topics made using the LDA algorithm. Section 5.3 presents the answers to RQ3, RQ4 and RQ5, which refer to the most used and least used terms. Section 5.4 presents the answers to RQ6 and RQ7 that analyze the answers' delay time after the question is asked and the duration of the discussion as well as the profile of the community members who asked questions and gave answers.

5.2 ANALYSIS OF RESEARCH QUESTIONS ONE AND TWO

Answers to these questions should start by revisiting the LDA model, which was presented in Section 2.5. It is a simple probabilistic procedure, in which a list of words must be provided by the researchers or generated automatically by the algorithm.

With the list completed, the LDA algorithm is used to find out the distribution of topics. To do this, it performs an inference in the probabilistic topic model: initially it tries to define the relevance of each word by the number of occurrences and the spreading within the body of the text object of the study. In this way, the algorithm creates an unsupervised classification and a set of grouped words is then generated. This set is called a topic (BLEI; NG; JORDAN, 2003).

Dirichlet distribution is used to show the topics' distribution. The Dirichlet sampling result is used to allocate words to different topics. We can then perceive the meaning of the name Latent Dirichlet Allocation, which expresses the intention of the model to allocate the latent topics that are distributed obeying the distribution of Dirichlet (BLEI; NG; JORDAN, 2003).

There are several possibilities of inference for LDA, the most used in academia is the Gibbs sampler, because it is a relatively simple implementation and it is appropriate for several problems. The Gibbs sampler is a Markov Chain Monte Carlo simulation capable of emulating high-dimensional probability distributions through the steady behavior of the Markov chain (HOFFMAN et al., 2013).

For better localization of terms, it is important to combine compound terms into a single sentence containing two to three consecutive words that often appear in the same order. For example, the term "conditional compilation" is grouped into the word conditionalcompilation. Likewise, "aspect oriented programming" becomes aspectorientedprogramming. This merge is indicated by an algorithm parameter.

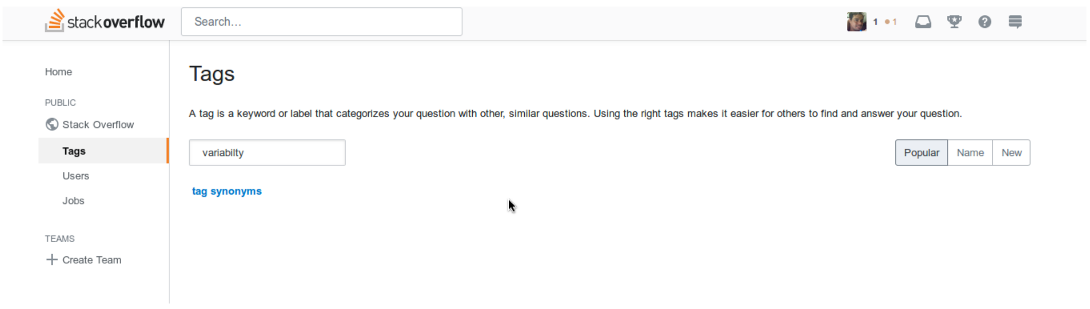
The LDA result is a list of words or terms grouped automatically by the algorithm. The identification of the subject or theme of each group of words is done by people who have knowledge of the specific domain. It may be very difficult to understand the meaning of the word list and to define titles that represent the subject they have in common when the thematic domain of the original text is not well known (HOFFMAN et al., 2013).

5.2.1 RQ1: What are the hot-topics that describe answered questions related to variability implementation mechanisms in Stack Overflow?

The procedure shown in figure 4.1 was used to answer this first research question. It is detailed next.

- As described in 4.3 Section, the Posts.xml file has been broken into two smaller files. One containing the questions and another containing the answers. From the file containing only the questions we separate the questions with total answers equal to or greater than one, thus defining a subset of questions that have answers.

  Unlike other search works in Stack Overflow, we were not able to use a subset of the dataset using searches based on tags related to our search themes. With few exceptions, like the word reuse, no identical or similar tags to the 52 words chosen were found. As an example, the figure 5.1 shows a search for the word "variability", which returned empty. Therefore, all retrieved datasets were used to generate two datasets with questions with at answer and with no answers.



Tags

A tag is a keyword or label that categorizes your question with other, similar questions. Using the right tags makes it easier for others to find and answer your question.

variabilty

Figure 5.1 Example of search for a tag in Stack overflow

- Next step was to remove the unnecessary signals and information from the XML file, leaving it with pure text, formed by the content of the title and the body of the question. For this purpose the set of stop words presented in appendix E was used.

- The LDA algorithm was configured following the recommendations of Linares-Vásquez, Dit e Poshyvanyk (2013). We chose these values because they represent the best LDA configuration for software engineering and because our interest focuses on high variability in the distribution of topics. The following parameters were used: number of documents = 11731053, topics = 30, interactions = 1000, $\alpha = 0.01$, $\beta = 0.01$ and Kappa learning parameter = 0.8, Kappa is a statistical index used to calculate agreement. When used in LDA in learning parameter, it guarantees a weighted result in the final evaluation for the topics.

- In order to easily identify the dominant topics, a list with 52 keywords that were used as an initial vocabulary by the LDA algorithm was used as a parameter to the LDA execution. This list can be seen in appendix 4 and the process of creating this vocabulary has been described in Section 4.3.

- Then we defined that the set of topics must have five words and we defined numerically an identification for the topics. As the list that we used as initial vocabulary has its words focused on variability, this was done according to the proposal of Allamanis e Sutton (2013).

- Next, the individual values of the words were summed up to obtain the total entropy value of the topic, as shown in the example shown in Table 5.1.

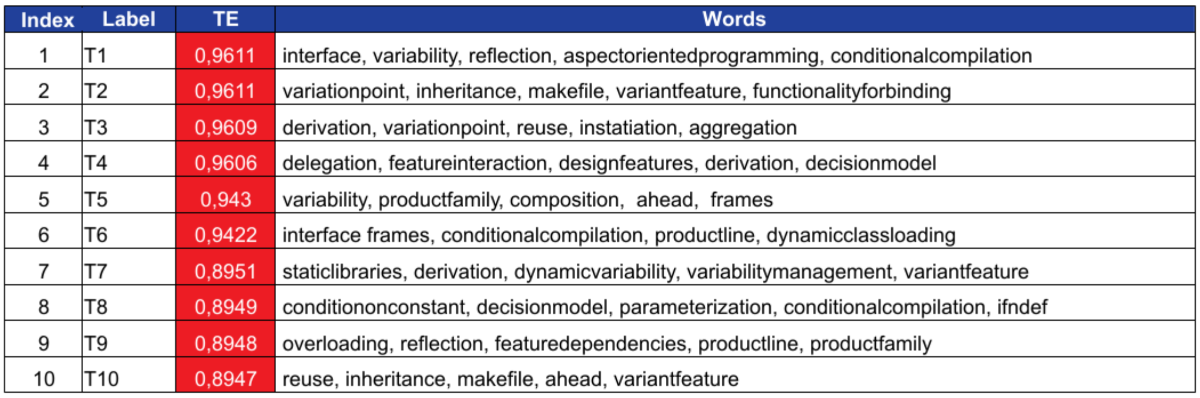
Table 5.1 Example of an LDA topic

Index	**Word**	**Entropy**
1	reuse	0,8757
2	inheritance	0,0058
3	makefile	0,0054
4	ahead	0,0052
5	variantfeature	0,0026
Total		0,8947

A subset containing 11,731,053 questions with answers was used. No previous selection of content was made, since a set of 52 words referring to the subject was used to feed the research. At this point we do not consider the answers because we understand that if the term is located in the question, its existence in the answer is redundant.

Since we already had a list of words, the identification of words became unnecessary. However, the purpose of this analysis is to validate the list of words previously defined by the experts, which will be used to answer questions 3 to 7.

Topics were organized in descending order according to their topical entropy metric (Topic Entropy (TE)) presented in Kappa: the higher the value, the greater the dispersion



Index	Label	TE	Words
1	T1	0,9611	interface, variability, reflection, aspectorientedprogramming, conditionalcompilation
2	T2	0,9611	variationpoint, inheritance, makefile, variantfeature, functionalityforbinding
3	T3	0,9609	derivation, variationpoint, reuse, instatiation, aggregation
4	T4	0,9606	delegation, featureinteraction, designfeatures, derivation, decisionmodel
5	T5	0,943	variability, productfamily, composition, ahead, frames
6	T6	0,9422	interface frames, conditionalcompilation, productline, dynamicclassloading
7	T7	0,8951	staticlibraries, derivation, dynamicvariability, variabilitymanagement, variantfeature
8	T8	0,8949	conditiononconstant, decisionmodel, parameterization, conditionalcompilation, ifndef
9	T9	0,8948	overloading, reflection, featuredependencies, productline, productfamily
10	T10	0,8947	reuse, inheritance, makefile, ahead, variantfeature

Figure 5.2 LDA Result for the dataset with questions answered

of the terms. This metric is expressed by the formula $TE(k) = -\sum_{i=1}^{|D|} \theta_{k,d}. \log(\theta_{k,d})$. For each topic, five representative words were selected.

Figure 5.2 presents the result obtained in the study, following the recommendations of Linares-Vásquez, Dit e Poshyvanyk (2013) and Barua, Thomas e Hassan (2014). With this we obtained 10 topics that represent the implementation of variability. Table 5.2 contains the words found in the topics. It is observed that of the 52 words from the input list, 33 were used, 16 appear only once and 14 appear two or three times.

It can be noticed that both, the words that occur most frequently in Table 5.2 and the first five topics in figure 5.2 (T1 to T5, with higher TE) are words that represent concepts related to reuse techniques that incorporate the concept of variability. Topic T3 for example contains the words reuse, instantiation, derivation, and variation point.

Continuing this analysis, it is noted that the topics T6, T7 and T8 contain words that refer to more specific techniques of variabilities' implementation: conditional compilation, ifndef, parameterization, reflection etc. Topics T9 and T10, with lower TE, contain words that combine concepts with implementation techniques.

The topics found confirm the experts' insights related to the problems that programmers face in implementing variability. The set of words located in the body of the texts of the questions studied through the use of the LDA algorithm allowed us to make discoveries that would not have been possible otherwise.

5.2.2 RQ2: What are the hot-topics that describe the unanswered questions related to variability implementation mechanisms in Stack Overflow?

This question is answered by following the steps used to answer RQ1 and changing only the body of text used, which in this case was the set of questions that did not get answers. A dataset with 1,741,743 questions was used. With the same set of words used to feed the LDA algorithm for RQ1, the results presented in figure 5.3 were obtained. Table 5.3 shows the words that appear in the topics as well as the counting of their frequency.

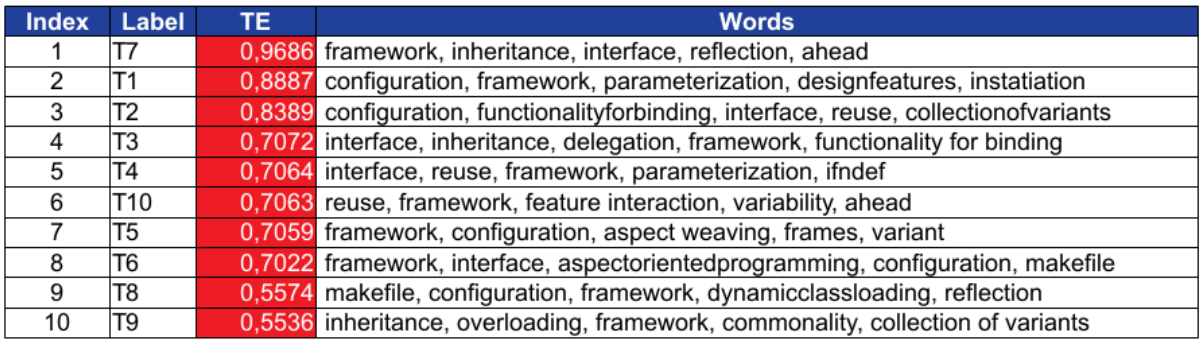
It can be noticed that the results are different, beginning with the fact that the TE parameter of the retrieved topics is on average inferior to the topics obtained in RQ1 and also that the number of words - 24 in total - is lower and there are some with higher

Table 5.2 Occurrence of words in the ten topics generated by LDA - RQ1

Index	**Word**	**Sum**
1	derivation	3
2	variant feature	3
3	conditional compilation	3
4	frames	2
5	interface	2
6	variability	2
7	reflection	2
8	inheritance	2
9	makefile	2
10	reuse	2
11	ahead	2
12	variation point	2
13	product family	2
14	product line	2
15	ifndef	1
16	instatiation	1
17	aggregation	1
18	delegation	1
19	composition	1
20	parameterization	1
21	overloading	1
22	variability management	1
23	dynamic variability	1
24	aspect oriented programming	1
25	feature modeling	1
26	feature dependencies	1
27	functionality for binding	1
28	static libraries	1
29	design features	1
30	condition on constant	1
31	dynamic class loading	1
32	feature interaction	1
Total		50

frequency, as framework (9), interface (5), configuration (6), and inheritance (3). They are considered to be more generic, occupy the first positions, and consume 44% of the 50 words in the list.

It is generally observed that the topics have a more general content, that is, they do not have content focused or specific to variabilities implementation. Topics T4 and T10 are



Index	Label	TE	Words
1	T7	0,9686	framework, inheritance, interface, reflection, ahead
2	T1	0,8887	configuration, framework, parameterization, designfeatures, instatiation
3	T2	0,8389	configuration, functionalityforbinding, interface, reuse, collectionofvariants
4	T3	0,7072	interface, inheritance, delegation, framework, functionality for binding
5	T4	0,7064	interface, reuse, framework, parameterization, ifndef
6	T10	0,7063	reuse, framework, feature interaction, variability, ahead
7	T5	0,7059	framework, configuration, aspect weaving, frames, variant
8	T6	0,7022	framework, interface, aspectorientedprogramming, configuration, makefile
9	T8	0,5574	makefile, configuration, framework, dynamicclassloading, reflection
10	T9	0,5536	inheritance, overloading, framework, commonality, collection of variants

Figure 5.3 LDA Result for the dataset with questions not answered

Table 5.3 Occurrence of words in the ten topics generated by LDA - RQ2

Index	Word	Sum
1	framework	9
2	interface	5
3	configuration	5
4	inheritance	3
5	reuse	3
6	reflection	2
7	ahead	2
8	parameterization	2
9	functionality for binding	2
10	collection of variants	2
11	makefile	2
12	design features	1
13	instatiation	1
14	variant	1
15	ifndef	1
16	feature interaction	1
17	variability	1
18	aspect weaving	1
19	frames	1
20	delegation	1
21	aspect oriented programming	1
22	dynamic class loading	1
23	overloading	1
24	commonality	1
Total		50

the ones that best contain words related to the implementation of variability. These topics indicate that the researched subject is also treated in the set of unanswered questions, that is, there are unanswered questions related to the implementation of variability, and with this we confirm the intuition of the specialists.

The Table 5.4 summarizes the words that appear in the two sets of retrieved topics and those that did not appear in any of them. The LDA, in general, assigns less weight to words with low frequency and those that always appear very close. These unused words are useful for answering to RQs 3, 4, and 5.

5.3 ANALYSIS OF RESEARCH QUESTIONS THREE TO FIVE

Answers to questions RQ3, RQ4 and RQ5 involved the initial execution of two main activities. The first one consisted in searching the original Posts.xml file retrieved from the Stack Overflow database, from the 13,472,784 existing questions, which could be used as the basis for the answers. Then the second activity comprised reading the recovered questions and removing those that dealt with other subjects. These two activities are presented in detail below.

5.3.1 Preparation

First, with the set of keywords that were validated by Rq1 and Rq2. And knowing that these keywords are closely related to the concept of software product lines, product families, and variabilities: they were used to retrieve the questions that contained them. For this, the methodology described in section ref methodology-2 was used. These words are listed in the first column of Table 5.5.

Then, the grep program for the Unix operating system was used to separate the questions from the answers. The total amount of questions retrieved is displayed in Table 5.5 in three columns. This Table displays question and answer totals for each word, the grand total of 2531 questions and 4262 answers, and a total of 6394 data points after duplications have been removed. It can be observed in Table 5.5 that the words “Conditional Compilation”, “Variability” and “Product Line” were the ones that had the highest number of questions and answers made by the community.

The “Stack Ov.” column contains the results obtained directly from the Stack Overflow site on January 27, 2018. When the value is higher in the Total column (eg, Feature Dependency), it indicates that questions have been removed by the community; when it is higher in the Stack Ov. column, it shows that the community continues to discuss the question.

Growth can be observed by the difference in values between the Total and Stack Ov columns. It is noteworthy, therefore, the great growth that there were recently of questions that contain the words “ifdef” and “ifndef”. These words represent implementation techniques in the context of conditional feedback.

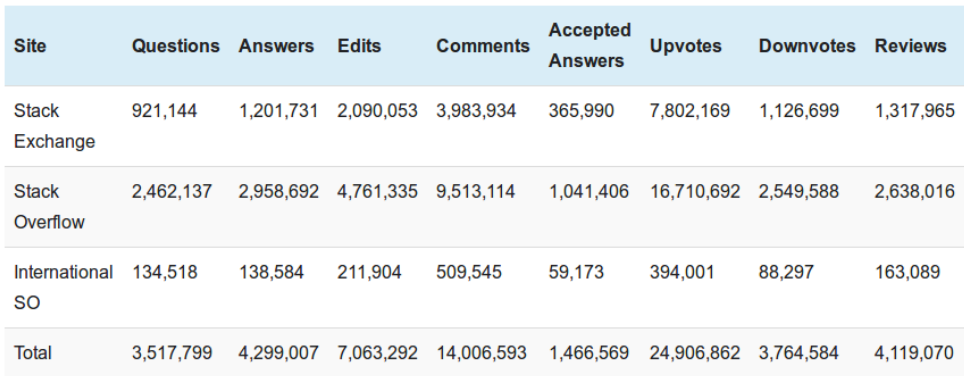
The Table shown in figure 5.4 presents some statistical data for the year of 2017, collected and published by the Stack Overflow site itself, where it can be noticed the expressive annual growth, with more than five million new questions and answers that year.

Table 5.4 Word localized or not localized in topic's retrieved for RQ1 and RQ2

Words used	**Words not used**
derivation	ifdef
framework	fop
configuration	foda
frames	decision model
interface	feature model
variability	feature ide
reflection	binding time
inheritance	condition on variable
makefile	modular feature
reuse	variability in space
ahead	operational dependencies
variant	constraint programming
ifndef	variability scope
instatiation	dynamic link libraries
aggregation	
delegation	
composition	
parameterization	
overloading	
commonality	
variation point	
product family	
product line	
conditional compilation	
variant feature	
variability management	
dynamic variability	
feature modeling	
feature dependencies	
feature interaction	
functionality for binding	
static libraries	
design features	
condition on constant	
dynamic class loading	
collection of variants	
aspect weaving	
aspect oriented programming	

Table 5.5 Results of the first retrieval

Word	Question	Answer	Total	Stack Ov
Commonality	283	642	925	946
Variability	443	874	1317	1381
Decision Model	6	8	14	16
Feature Interaction	1	3	4	7
Feature Dependencies	21	9	30	26
Feature Model	33	18	51	63
FOP (Feat. Orient. Prog.)	2	3	5	8
Product Line	351	306	657	582
Product family	124	73	197	158
Variation Point	4	15	19	16
Conditional Compilation	750	1832	2582	2735
ifdef	312	281	593	2415
ifndef	201	198	399	946
Total	2531	4262	6394	9301



Site	Questions	Answers	Edits	Comments	Accepted Answers	Upvotes	Downvotes	Reviews
Stack Exchange	921,144	1,201,731	2,090,053	3,983,934	365,990	7,802,169	1,126,699	1,317,965
Stack Overflow	2,462,137	2,958,692	4,761,335	9,513,114	1,041,406	16,710,692	2,549,588	2,638,016
International SO	134,518	138,584	211,904	509,545	59,173	394,001	88,297	163,089
Total	3,517,799	4,299,007	7,063,292	14,006,593	1,466,569	24,906,862	3,764,584	4,119,070

Figure 5.4 Statistics of Stack Overflow – 2017

5.3.2 Removal of questions not related to the topics searched

In order to guide the research, the resulting questions were read throughout their contents (Body and Title fields) with the objective of eliminating possible questions that were not directly related to the study, as presented in the methodology described in Section 4.1. The result of the removals can be seen in Table 5.6. It can be noticed that the final number or questions retrieved was 1962. The years that these questions have been asked is shown in Table 5.7. After some growth until 2012, there was a stabilization in the number of questions. Note that the 2017 total is partial.

The word Variability was the one that had the highest number of removals, with a total of 237 questions. One of the main reasons for the removals was due to the fact that several questions dealt with variations related to processing times. The terms Product Line and Product Family had questions removed because many of them dealt with information

Table 5.6 Results of Question's removal

Word	**Question**	**Removed**	**Final Total**
Commonality	283	67	216
Variability	443	237	206
Decision Model	6	1	5
Feature Interaction	1	0	1
Feature Dependencies	21	0	21
Feature Model	33	0	33
FOP (Feat. Orient. Prog.)	2	0	2
Product Line	351	101	250
Product Family	124	51	73
Variation Point	4	1	3
Conditional Compilation	750	38	712
ifdef	312	54	258
ifndef	201	19	182
Total	2531	569	1962

Table 5.7 Questions retrieved by year

Year	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
Total	20	95	150	214	233	340	273	290	278	69

in databases related to information systems. The words Feature Interaction, Feature Dependencies, Feature Model, and Feature Oriented Program have not had any questions removed since they all address the subject of this research.

5.3.3 RQ3 - Do mechanisms of variability exist that are discussed by the Stack overflow community?

Considering Table 5.8, it is noted that even though the research topic has not had a large number of discussions in the Stack Overflow community, the number of questions retrieved shows that the community is attentive to the topic. Additionally, some themes arouse much more interest than others. It should be considered that the Stack Overflow community has its focus on development and with that it emphasizes a large number of more technical subjects related to programming languages, databases, operating systems and the whole set of associated technologies and tools to this environment.

Selected questions, with a total of 1962, were then analyzed quantitatively as to the repercussion they had among Stack Overflow users. The results were initially separated into questions, answers and accepted answers. It should be noted that, on average, the number of answers was 2.2 times higher than the number of questions (or 220 % higher). Subjects with few questions also had proportionally few answers, with one exception for Variation Point, which had 15 answers for 3 questions. The total mean number of answers accepted was 0.58, thus indicating that 58 % of the questions have answer accepted by the community.

Table 5.8 Repercussion of answers

Word	Questions	Answers	An/Q	Accepted	Ac/Q
Commonality	216	642	3,0	119	0,6
Variability	206	874	4,2	111	0,5
Decision Model	5	8	1,6	3	0,6
Feature Interaction	1	3	3,0	0	0
Feature Dependencies	21	9	0,4	7	0,3
Feature Model	33	18	0,5	21	0,6
FOP (Feat. Orient. Prog.)	2	3	1,5	1	0,5
Product Line	250	306	1,2	120	0,5
Product family	73	73	1,0	33	0,5
Variation Point	3	15	5,0	0	0
Conditional Compilation	712	1832	2,6	458	0,6
ifdef	258	281	1,1	145	0,6
ifndef	182	198	1,1	117	0,6
Total	1962	4262	2,2	1135	0,58

Some themes, represented by the words Variability, Commonality and Conditional Compilation, had the highest number of questions and also received a higher average number of answers: 4.2, 3.0 and 2.6, respectively. This characterizes them as themes of interest.

The average of the accepted questions is fairly homogeneous (or with little dispersion) among the selected words. This indicates that approximately 50-60% of the answers were accepted by the questioner for all words, except for three that almost did not receive questions or answers (Feature Interaction, FOP, Variation Point).

The average numbers of answers and accepted answers are respectively 1.58 and 0.60. When compared to the averages of the entire retrieved dataset, shown in Table 5.8, these numbers show that the repercussion of the questions related to the words retrieved in this survey are slightly larger than the overall average. Therefore, it can be said that the repercussion of these themes is very good, compatible with the general average repercussion of the site.

Going deeper into the analysis, Table shown in Figure 5.9 shows other aspects of the Stack Overflow community interaction with the selected questions: the number of visits the questions and answers received. When a theme is searched in Stack Overflow, the question that answers the search is shown to the user and just below the answers appear. Therefore, the number of visits is considered for both, questions and answers, and not separately.

Note that the absolute value of visits is high for most words, as well as their average value. The words with the highest average number of visits are Conditional Compilation,

Table 5.9 Results of visits

Word	**Questions**	**Visits**	**Average**
Commonality	216	272.495	1.261,55
Variability	206	252.197	1.224,26
Decision Model	5	803	160.6
Feature Interaction	1	726	726
Feature Dependencies	21	11.359	540,90
Feature Model	33	22.011	667
FOP (Feat. Orient. Prog.)	2	3.400	1.700
Product Line	250	307.182	1.228,73
Product Family	73	203.040	2.781,33
Variation Point	3	386	128,67
Conditional Compilation	712	1.313.943	1.845,43
ifdef	258	416.267	1.613,44
ifndef	182	340.314	1.869,86
Total	1.962	3.144.123	1.602,51

ifdef, ifndf, and Product Line, in this order. The overall average of visits to the questions retrieved in the original dataset is 1.815,16. Therefore, the average of 1,602.5 visits these questions received are slightly lower than the overall average of the site. This allow us to affirm that our research topic (variability implementation) has a good impact on Stack Overflow, close to the average impact of other questions.

Any user registered in Stack Overflow can vote on a question only once or, in other words, if answer has 5,105 votes in total this means that this is the total number of users who have accessed the questions and voted positively for them. Usually users vote when a question (or even some answer) had value for him or her. To vote it is necessary that the user has accumulated more than 300 points of participation in the site.

There is also the possibility for a user to indicate that a question is his or hers favorite. Favorite works as a "like" in social networks and it is an action reserved for those who have less than 300 participation points on the site and also for unregistered visitors. It is therefore a more informal manifestation than the vote.

The Tables 5.10 and 5.11 show the averages of votes and favorites. We highlight that Conditional Compilation and Commonality had the most expressive averages of votes and that the proportion of votes per question is 2.66. Conditional Compilation and Product Family were the words that appear as favorites with more expressive number and the proportion of favorites by questions is 0,84, which is higher than the average proportion 0,58 of the whole dataset (shown in Table 5.8).

It is interesting to observe that two more general words, such as Commonality and Product Family, also stand out in this analysis. As in the previous analysis, there are words that have received little attention, such as Feature Interaction, FOP, Variation Point, and Decision Model. Similarly to the other cases, the votes and the favorites also indicate that there is discussion and repercussion of the themes researched in Stack

Table 5.10 Results of votes

Word	Questions	Votes	Average
Commonality	216	645	2,99
Variability	206	353	1,71
Decision Model	5	2	0,4
Feature Interaction	1	1	1
Feature Dependencies	21	39	1,86
Feature Model	33	33	1
FOP (Feat. Orient. Prog.)	2	16	8
Product Line	250	358	1,43
Product Family	73	140	1,92
Variation Point	3	4	1,33
Conditional Compilation	712	2.648	3,72
ifdef	258	592	2,29
ifndef	182	385	2,12
Total	1.962	5.105	2,66

Overflow.

Table 5.11 Results of favorite

Word	Questions	Favorite	Average
Commonality	216	195	0,90
Variability	206	104	0,50
Decision Model	5	5	1
Feature Interaction	1	0	0
Feature Dependencies	21	15	0,71
Feature Model	33	37	1,2
FOP (Feat. Orient. Prog.)	2	3	1,5
Product Line	250	130	0,52
Product Family	73	79	1,08
Variation Point	3	1	0,33
Conditional Compilation	712	861	1,21
ifdef	258	139	0,54
ifndef	182	85	0,47
Total	1.962	1.654	0,84

In terms of averages, the 1962 questions related to our search received 2.66 votes, which is higher than the average of votes from the dataset: 1,85 (see Table 5.8). The same occurred with the average of votes, 0.84, which is much higher than the average of the dataset: 0,04. These data show the good repercussion of questions related to variability implementation.

Table 5.12 Question and Answer for the data set of words

Classification	**Total**	**Percentage**	**Percentage**
Questions	1.962	37,03%	
Questions with accepted answers	(1.135)		57,85%
Questions without accepted answers	(827)		42,15%
Answers	3.337	62,97%	
Total	5.299	100.00%	100%

Table 5.13 Averages of Answers

	Average
Answers per Question (total)	1,70
Answers Accepted per Question	0,58

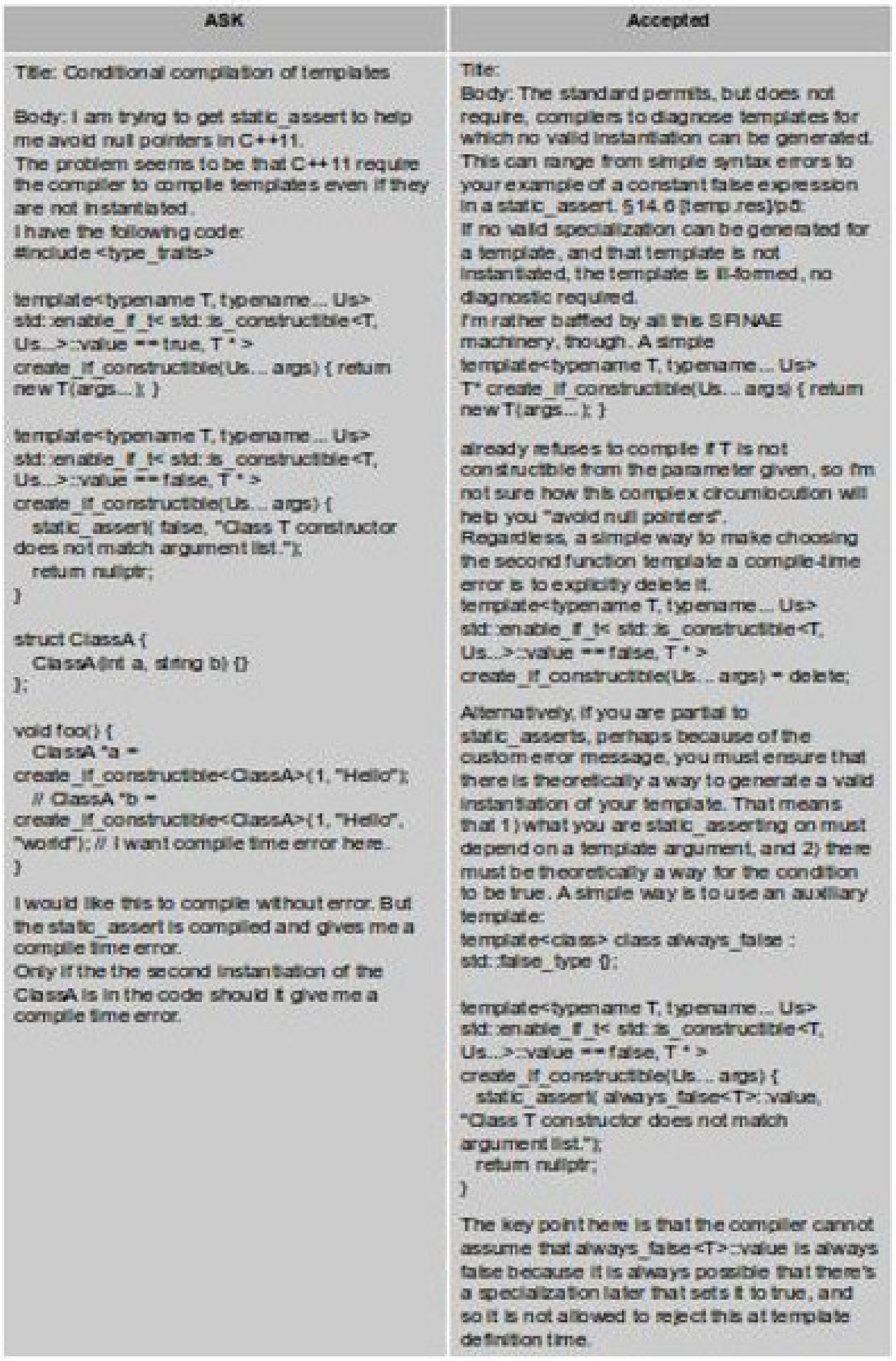
The Table 5.12 shows the percentages of answers and accepted answers for the subset of 1962 selected questions. The averages of questions with answers and questions with accepted answers, respectively of 1.70 and 0.58, are presented in Table 5.13. These numbers are slightly higher than those collected for the whole retrieved dataset presented in Table 5.13: 1.58 for the answers and 0.54 for the accepted answers. This, again shows the repercussion of the searched words is compatible with the overall impact of the site.

An example of a question with an accepted answer is shown in Figure 5.5. The question refers to a doubt about Conditional Compilation and includes code mixed with text. The answer, on the right side of Table, does not have a title, since all answers are stored with reference to the question record, as explained in Section 4.2

The term Product line presents a total of 250 questions for 360 answers, that is, the number of answers is 44 % greater than that of questions. The column of accepted answers presents 120 answers, that is, 44 % of the total of questions have some answer from the community. An example of a question related to this word is shown in Figure 5.6, where a software engineer seemingly beginning to develop software product lines looks for general information about the subject, especially about topics such as configuration, development cycle, and testing.

Two other product line questions are presented in Figure 5.7. The question on the left side shows a programmer who realizes the possibility of reusing an API for two different situations and wants to use a solution based on the concept of variations in product lines. In the right-hand question the programmer is doing maintenance for a product line and has a similar doubt.

Questions listed in Figure 5.8 are related to (Conditional Compilation (CC) Conditional Compilation. In both questions the programmers want options to optionally take some code snippets defined at compile time to the final code and want to use conditional compilation to achieve this. It is unclear whether they are working with Software Product Lines or have had perceived a reuse opportunity. The question shown in figure 5.9 refers as well to the use of Conditional Compilation, but this time to implement the Model,



ASK

Title: Conditional compilation of templates

Body: I am trying to get static_assert to help me avoid null pointers in C++11.
The problem seems to be that C++11 require the compiler to compile templates even if they are not instantiated.
I have the following code:

```
#include <type_traits>

template<typename T, typename... Us>
std::enable_if_t< std::is_constructible<T,
Us...>::value == true, T * >
create_if_constructible(Us... args) { return
new T(args...); }

template<typename T, typename... Us>
std::enable_if_t< std::is_constructible<T,
Us...>::value == false, T * >
create_if_constructible(Us... args) {
   static_assert( false, "Class T constructor
does not match argument list.");
   return nullptr;
}

struct ClassA {
   ClassA(int a, string b) {}
};

void foo() {
   ClassA *a =
create_if_constructible<ClassA>(1, "Hello");
   // ClassA *b =
create_if_constructible<ClassA>(1, "Hello",
"world"); // I want compile time error here.
}
```

I would like this to compile without error. But the static_assert is compiled and gives me a compile time error.
Only if the the second instantiation of the ClassA is in the code should it give me a compile time error.

Accepted

Title:
Body: The standard permits, but does not require, compilers to diagnose templates for which no valid instantiation can be generated. This can range from simple syntax errors to your example of a constant false expression in a static_assert. §14.6 [temp.res]/p8:
If no valid specialization can be generated for a template, and that template is not instantiated, the template is ill-formed, no diagnostic required.
I'm rather baffled by all this SFINAE machinery, though. A simple

```
template<typename T, typename... Us>
T* create_if_constructible(Us... args) { return
new T(args...); }
```

already refuses to compile if T is not constructible from the parameter given, so I'm not sure how this complex circumlocution will help you "avoid null pointers".
Regardless, a simple way to make choosing the second function template a compile-time error is to explicitly delete it.

```
template<typename T, typename... Us>
std::enable_if_t< std::is_constructible<T,
Us...>::value == false, T * >
create_if_constructible(Us... args) = delete;
```

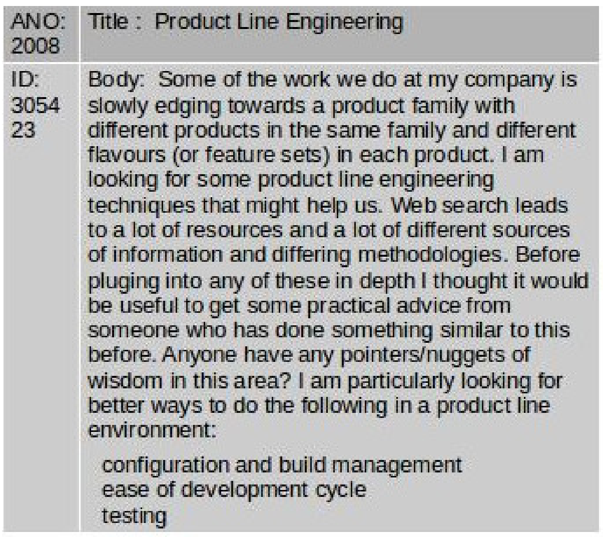
Alternatively, if you are partial to static_asserts, perhaps because of the custom error message, you must ensure that there is theoretically a way to generate a valid instantiation of your template. That means that 1) what you are static_asserting on must depend on a template argument, and 2) there must be theoretically a way for the condition to be true. A simple way is to use an auxiliary template:

```
template<class> class always_false :
std::false_type {};

template<typename T, typename... Us>
std::enable_if_t< std::is_constructible<T,
Us...>::value == false, T * >
create_if_constructible(Us... args) {
  static_assert( always_false<T>::value,
"Class T constructor does not match
argument list.");
  return nullptr;
}
```

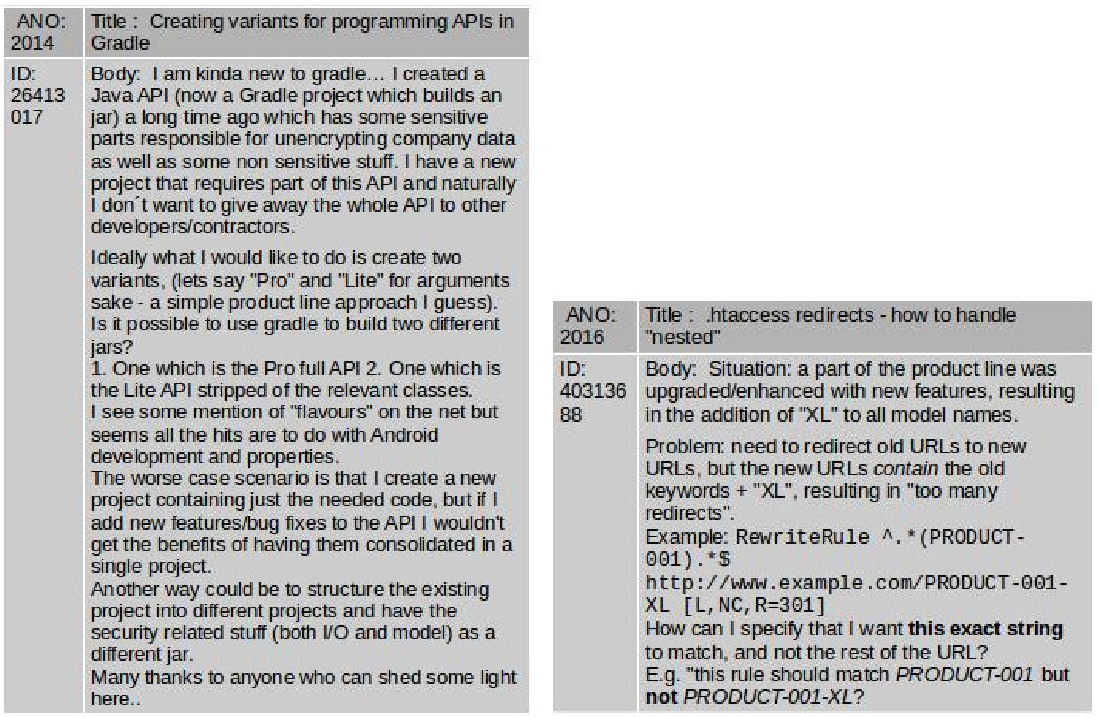
The key point here is that the compiler cannot assume that always_false<T>::value is always false because it is always possible that there's a specialization later that sets it to true, and so it is not allowed to reject this at template definition time.

Figure 5.5 Question with an accepted answer



ANO: 2008	Title : Product Line Engineering
ID: 305423	Body: Some of the work we do at my company is slowly edging towards a product family with different products in the same family and different flavours (or feature sets) in each product. I am looking for some product line engineering techniques that might help us. Web search leads to a lot of resources and a lot of different sources of information and differing methodologies. Before pluging into any of these in depth I thought it would be useful to get some practical advice from someone who has done something similar to this before. Anyone have any pointers/nuggets of wisdom in this area? I am particularly looking for better ways to do the following in a product line environment: configuration and build management ease of development cycle testing

Figure 5.6 A question related to Product Line

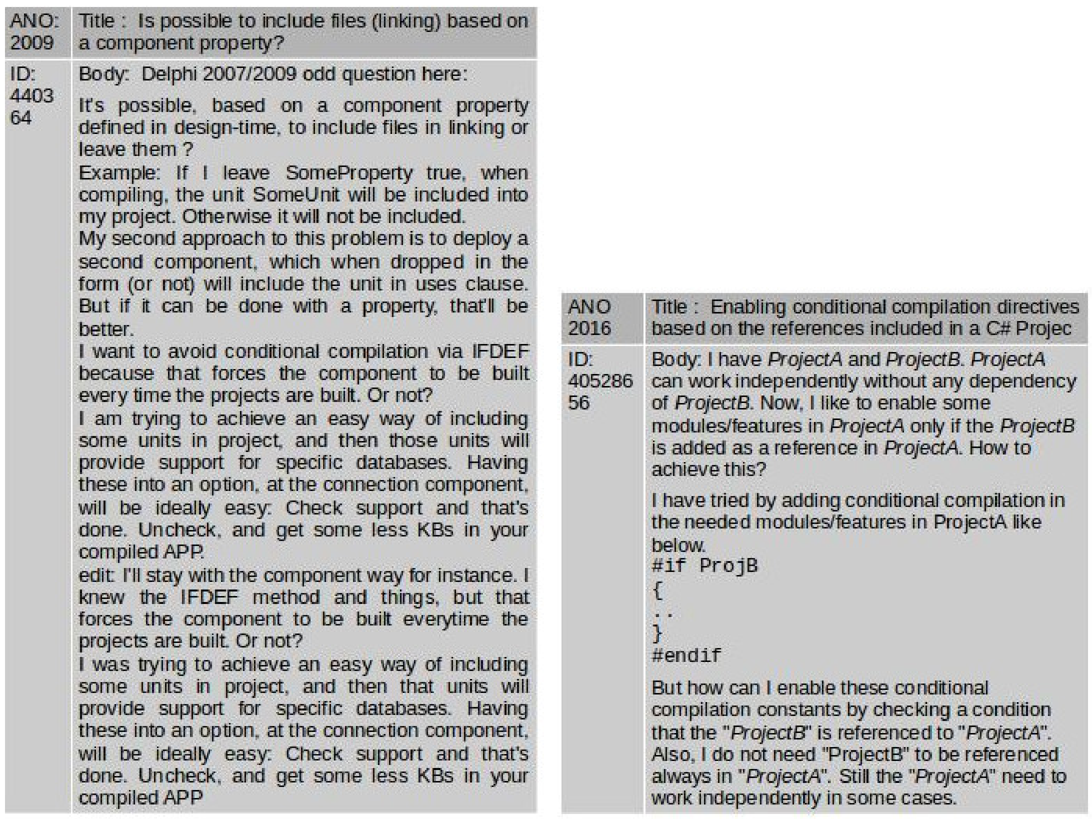


ANO: 2014	Title : Creating variants for programming APIs in Gradle
ID: 26413017	Body: I am kinda new to gradle… I created a Java API (now a Gradle project which builds an jar) a long time ago which has some sensitive parts responsible for unencrypting company data as well as some non sensitive stuff. I have a new project that requires part of this API and naturally I don´t want to give away the whole API to other developers/contractors. Ideally what I would like to do is create two variants, (lets say "Pro" and "Lite" for arguments sake - a simple product line approach I guess). Is it possible to use gradle to build two different jars? 1. One which is the Pro full API 2. One which is the Lite API stripped of the relevant classes. I see some mention of "flavours" on the net but seems all the hits are to do with Android development and properties. The worse case scenario is that I create a new project containing just the needed code, but if I add new features/bug fixes to the API I wouldn't get the benefits of having them consolidated in a single project. Another way could be to structure the existing project into different projects and have the security related stuff (both I/O and model) as a different jar. Many thanks to anyone who can shed some light here..

ANO: 2016	Title : .htaccess redirects - how to handle "nested"
ID: 40313688	Body: Situation: a part of the product line was upgraded/enhanced with new features, resulting in the addition of "XL" to all model names. Problem: need to redirect old URLs to new URLs, but the new URLs *contain* the old keywords + "XL", resulting in "too many redirects". Example: `RewriteRule ^.*(PRODUCT-001).*$ http://www.example.com/PRODUCT-001-XL [L,NC,R=301]` How can I specify that I want **this exact string** to match, and not the rest of the URL? E.g. "this rule should match *PRODUCT-001* but **not** *PRODUCT-001-XL*?

Figure 5.7 Examples of Questions related to Variability

View, Controller (MVC) pattern.



ANO: 2009	Title : Is possible to include files (linking) based on a component property?
ID: 4403 64	Body: Delphi 2007/2009 odd question here: It's possible, based on a component property defined in design-time, to include files in linking or leave them ? Example: If I leave SomeProperty true, when compiling, the unit SomeUnit will be included into my project. Otherwise it will not be included. My second approach to this problem is to deploy a second component, which when dropped in the form (or not) will include the unit in uses clause. But if it can be done with a property, that'll be better. I want to avoid conditional compilation via IFDEF because that forces the component to be built every time the projects are built. Or not? I am trying to achieve an easy way of including some units in project, and then those units will provide support for specific databases. Having these into an option, at the connection component, will be ideally easy: Check support and that's done. Uncheck, and get some less KBs in your compiled APP. edit: I'll stay with the component way for instance. I knew the IFDEF method and things, but that forces the component to be built everytime the projects are built. Or not? I was trying to achieve an easy way of including some units in project, and then that units will provide support for specific databases. Having these into an option, at the connection component, will be ideally easy: Check support and that's done. Uncheck, and get some less KBs in your compiled APP

ANO 2016	Title : Enabling conditional compilation directives based on the references included in a C# Projec

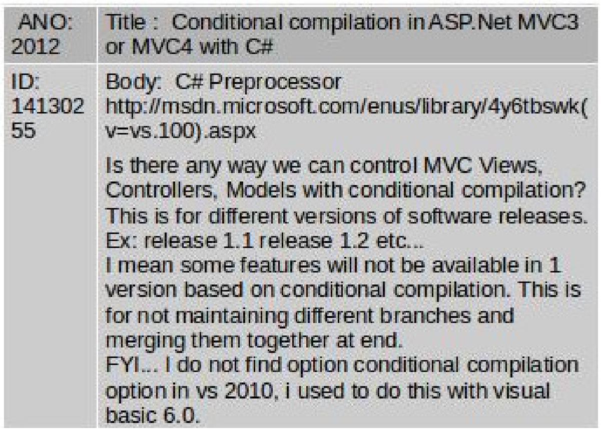
ID: 405286 56	Body: I have *ProjectA* and *ProjectB*. *ProjectA* can work independently without any dependency of *ProjectB*. Now, I like to enable some modules/features in *ProjectA* only if the *ProjectB* is added as a reference in *ProjectA*. How to achieve this?
	I have tried by adding conditional compilation in the needed modules/features in ProjectA like below.
	`#if ProjB`
	`{`
	`..`
	`}`
	`#endif`
	But how can I enable these conditional compilation constants by checking a condition that the "*ProjectB*" is referenced to "*ProjectA*". Also, I do not need "ProjectB" to be referenced always in "*ProjectA*". Still the "*ProjectA*" need to work independently in some cases.

Figure 5.8 Example of two questions related do Conditional Compilation

In conclusion, the analysis presented in this Section shows that there are discussions related to the implementation of variability in the form of answers to the questions, and actions to accept and register answer as a favorite. It is therefore possible to answer positively to question RQ3. This statement is based on the results presented, totalizing 4262 answers, which added to the 1962 questions result in 6224 lines; in the quantity of answers given and answers considered accepted; and the number of votes, marking as favorite and visits. However, it is a small community, considering the whole of the Stack Overflow community. It is also noted that in some questions the programmer has a reuse problem that can be solved using concepts and techniques of implementation of variabilities, but seems to be unaware of this.

We can also conclude that the internal and external community of Stack overflow is researching the subject of implementation of variability, represented in our study by the questions related to the searched words. This can be proven by ascertaining the repercussion of the questions in the form of answers, accepted answers, views, votes and favorites, which are in the average of the site or slightly above in some cases.

Some characteristics of Stack Overflow indicate that the information presented in the community follows a continuous audit process, so that the quality of the information allows us to rely on the discussions regarding the implementation of variability that occur



ANO: 2012	Title : Conditional compilation in ASP.Net MVC3 or MVC4 with C#
ID: 141302 55	Body: C# Preprocessor http://msdn.microsoft.com/enus/library/4y6tbswk(v=vs.100).aspx Is there any way we can control MVC Views, Controllers, Models with conditional compilation? This is for different versions of software releases. Ex: release 1.1 release 1.2 etc... I mean some features will not be available in 1 version based on conditional compilation. This is for not maintaining different branches and merging them together at end. FYI... I do not find option conditional compilation option in vs 2010, i used to do this with visual basic 6.0.

Figure 5.9 Question about using CC to implement MVC

in the site. This is due to the implemented system, in which the community itself manages itself through votes and through mediators, who are experienced users with high scores, with the power to remove questions or answers, when they are not appropriate or when they are repeated.

Finally, since the beginning of Stack overflow activities in 2008, issues of implementation of variability have been discussed by the community. Stack Overflow users do not always clearly describe the problem in question and the title formulation. In most cases, the questions are accompanied by examples of source code texts. Most likely for all these reasons and maybe others, some questions do not have an answer.

5.3.4 RQ4 - What are the techniques related to variability implementation mechanisms most discussed in Stack overflow?

To ensure greater breadth of study and better answer this question, a set of words more related to techniques of implementation of variabilities was chosen, expanding those already existing words in the initial set. Experts contributed to this group, but articles dealing with the implementation of variability were also consulted. These words are presented in the first column of Table 5.14.

With this list of words, the new words were searched in the set of 1962 selected questions and the old ones had already been searched and maintained the same results, which is presented in Table 5.6. This set is known as a set of words that refer to terms (or mechanisms) related to the implementation of variabilities.

Considering the results obtained, the words were grouped into three groups, as shown in Table 5.14. The first group presents the words with more than one hundred questions, the second group contains the words with ten to one hundred questions, and the third group contains the words with less than ten questions.

Group one is the one with the most questions. According to the analysis of RQ3, these questions in general are the ones that had the most answers, visits, votes and nominations of favorite. These are, therefore, the most discussed mechanisms in Stack Overflow related to the implementation of variabilities. The second group presents a set

Table 5.14 Frequency of variability implementation words

Word	**Total**
Conditional Compilation	712
ifdef	312
Version	281
ifndef	201
Configuration	125
Interface	93
Framework	82
Service	60
Inheritance	29
Reuse	17
Reflection	17
Instatiation	12
Frames	10
Overloading	8
Ahead	7
Aggregation	3
Composition	3
Delegation	1
Parameterization	0
Dynamic class loading	0
Static libraries	0
Dynamic link libraries	0
Aspect oriented programming	0
Aspect weaving	0
Derivation	0

of words that received a low number of questions regarding the first group and which in themselves would not allow a positive answer to RQ4. The third group contains words that received very few questions or no questions. This group is discussed in the answer to question RQ5.

In order to better understand these sets of words, they were classified into three categories representing more general concepts: pre-processing, versioning and reuse. The classification is displayed in Table 5.15. The third group will be detailed in the next subsection to answer question RQ5.

Group one contains words related to the category of preprocessors and version control. In the first category the words Configuration, Version and Conditional Compilation appear. The most known variability implementation mechanisms are those related to preprocessors that use commands like ifdef and ifndef to conditionally compile variable features. This mechanism is used in many popular software, such as the Linux operating

Table 5.15 Groups of frequency

Group	**Pre-processing**	**Version Control**	**Reuse**
1	Conditional Compilation	version	
	ifdef	Configuration	
	ifndef		
2	Reflection	Instantiation	Service
	frames		Reuse
			Framework
			Interface
			Inheritance
3	Ahead		Overloading
	Parameterization		Agregation
	Dynamic class loading		Delegation
	Static libraries		
	Composition		
	Dynamic link libraries		
	Aspect weaving		
	Aspect oriented programming		
	Derivation		

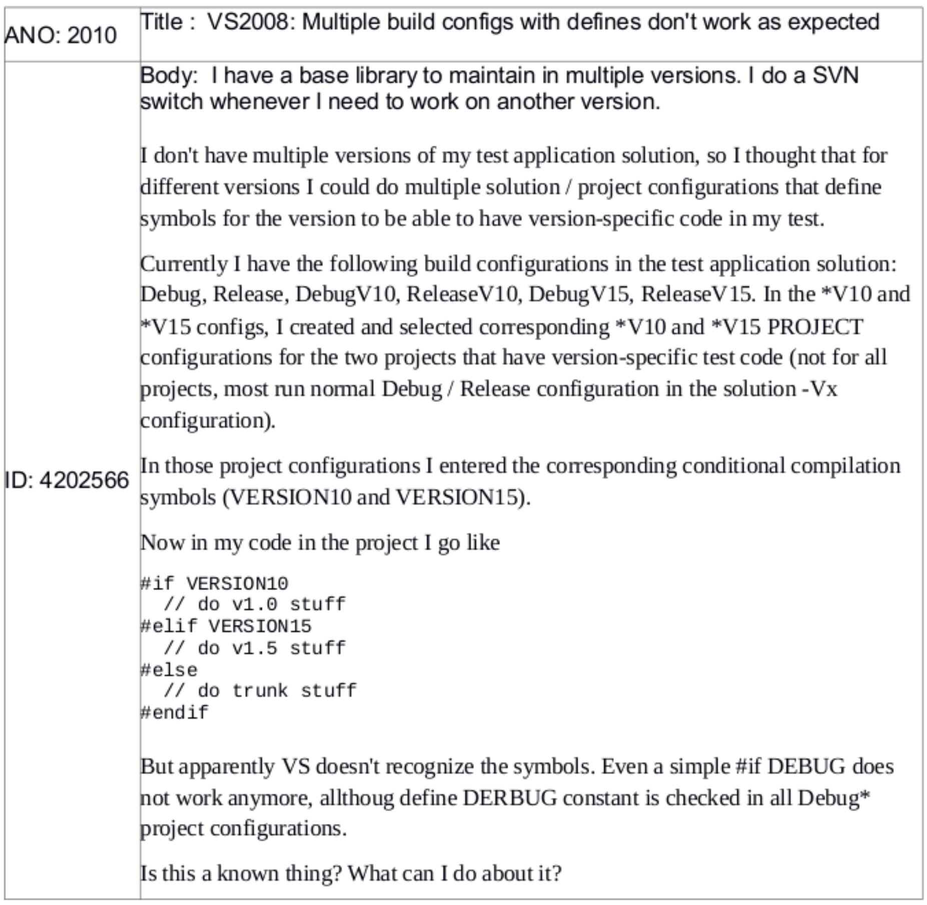
system. The category of words related to version control can also be observed with the words Version and Configuration, which appear with a very significant number of questions from group one. These concepts are related to the construction of new software products with a complex set of variations.

An example question that illustrates well this set of words is presented in Table 5.10. In it, the words Version, Configuration and Conditional Compilation appear. The code snippet example shown uses an IF command, but could have been more appropriately an ifdef.

As well as the 5.7 and 5.8 questions shown in RQ3, the programmer who asked the question has a "reuse-in-small" problem: the need to manage multiple versions of a library. He seems to have notion that conditional compilation may be a solution to his problem but, by the question, does not demonstrate to have more in-depth theoretical knowledge on the conditional compilation mechanisms and on the concept of variability and other related concepts such as software product lines and code generation.

The second group of words is more concentrated in the reuse category. Words in this category are general reuse mechanisms that can also be used to implement variabilities, such as services, frameworks, and object oriented mechanisms (inheritance, overloading). They are more complex implementation mechanisms and generally used in specific application domain well defined. The word Reflection also represents a complex and very specialized concept. This in a way explains why these terms have fewer questions and are less commonly used by the Stack Overflow community.

Word Instantiation in the Version Control category also appears in this group and is closely related to configuration. Our hypothesis is that it has fewer questions and is less used because the problems reported, such as in the 5.10 question, are very basic and programmers do not seem to know the abstract concept of generalizing in advance the solution of a problem with many variables and use the technology to "instantiate" the appropriate solution (or product) when the problem appears.



ANO: 2010

Title : VS2008: Multiple build configs with defines don't work as expected

ID: 4202566

Body: I have a base library to maintain in multiple versions. I do a SVN switch whenever I need to work on another version.

I don't have multiple versions of my test application solution, so I thought that for different versions I could do multiple solution / project configurations that define symbols for the version to be able to have version-specific code in my test.

Currently I have the following build configurations in the test application solution: Debug, Release, DebugV10, ReleaseV10, DebugV15, ReleaseV15. In the *V10 and *V15 configs, I created and selected corresponding *V10 and *V15 PROJECT configurations for the two projects that have version-specific test code (not for all projects, most run normal Debug / Release configuration in the solution -Vx configuration).

In those project configurations I entered the corresponding conditional compilation symbols (VERSION10 and VERSION15).

Now in my code in the project I go like

```
#if VERSION10
  // do v1.0 stuff
#elif VERSION15
  // do v1.5 stuff
#else
  // do trunk stuff
#endif
```

But apparently VS doesn't recognize the symbols. Even a simple #if DEBUG does not work anymore, allthoug define DERBUG constant is checked in all Debug* project configurations.

Is this a known thing? What can I do about it?

Figure 5.10 Question referring to several keywords

As we can see, the variability implementation mechanisms have received the attention of the Stack Overflow community and the words presented in group one of Table 5.15 represent the most discussed and most widely used variability implementation mechanisms.

5.3.5 RQ5 - What are the mechanisms of variability implementation not discussed by the Stack overflow community?

As a first part of the answer to this question, it is possible to state that the words in group 3 of Table 5.14 do not refer to mechanisms discussed by the Stack Overflow community. It should be emphasized, however, that the results of the searches performed are restricted to the 1962 selected questions and contain other words related to variabilities. Some of these terms have other technical and colloquial meanings and if they are searched in isolation directly from the Stack Overflow site, many questions will be recovered, but outside the scope of variability implementation mechanisms.

To complete the answer to this question we used another list of words forming terms or phrases related to mechanisms of variability, which were collected from articles by Svhnbert et al. and Gacej and Anastasopoules (SVAHNBERG; GURP; BOSCH, 2005) (GACEK; ANASTASOPOULES, 2001), and supplemented with other words and terms proposed by the experts. This new list was used as input for a search of the original dataset and also in the Stack Overflow search tool, using quotation marks to get exact results for the search.

As a result of the search, a list of terms related to variability implementation mechanisms for which no questions were found have been obtained. This list is displayed in Table 5.16. We can also say that this set represents words and terms related to mechanisms of variabilities implementation that are not discussed by the Stack Overflow community.

The terms not found in the discussions, which are well known in academia, do not necessarily represent variability implementation techniques that are not used in practice. They simply have not been located in the Stack overflow data base. One possible reason for this may be that the community has its focus on code building as we have seen, and most of these terms are not directly related to the programming activity but are rather more conceptual and related to system design, program design, and other phases of a software development process.

5.4 ANALYSIS OF RESEARCH QUESTIONS SIX AND SEVEN

5.4.1 RQ6 - How fast is the crowd at covering widely variability implementation mechanisms?

To answer the RQ6 question we computed the dates and times of the 1962 questions selected to answer question RQ3 and also the dates and times of their answers to calculate how long the stack overflow community takes to react to an issue. Once the questions and their respective answers were selected, a file was created in a relational database. Then, the date and time data were stored in separate columns to facilitate calculation.

Notice taht the item creation date formation (question and answer) was separated because in the original XML format the date and the time form a single string and are separated by the letter ”T”, as can be seen in the text example of the Post.xm file contained in Chapter 4 (Example: “ CreationDate = 2008-08-01T12: 35 : 56917”).

As an example of the calculation, the data retrieved from some questions are listed below:

```
< conditional >
Pergunta id: 60673, total de respostas: 22
Tempo para primeira resposta: 0:20 h.
Tempo para ultima resposta maior ou igual a: 6 Anos
```

Table 5.16 list of words not discussed

Keyword
Variability Identification
Variability Modeling
Product Derivation
Variability Evolution
Feature-oriented domain analysis
Feature interaction and dependencies
mced
Optimization of variability
Multi-level feature trees
Software Variability
Product Architecture Derivation
Condition on Variable
Code Fragment Superimposition
Variant Component Specializations
Optional Component Specializations
Variant Component Implementations
Code Fragment Superimposition
Feature-binding units
Runtime variation points
Run-time variation points
Feature Model Analyzer
Product Line Unified Modeller
Crosscutting feature
Operational feature dependency
Separation of Feature Dependencies
Capacitive features
Resistive features
Variability in space
Variability Constraints
Variability Scope
Feature Binding Units
Modification of Variants
Modification of Variation Points
Static variability models
Variability Implementation
Product derivation

```
Pergunta id: 71277, total de respostas: 3
Tempo para primeira resposta: 7:53 h.
Tempo para ultima resposta: 22 Dias e 21:33 h.

Pergunta id: 97114, total de respostas: 6
Tempo para primeira resposta: 0:5 h.
Tempo para ultima resposta: 1 Dias e 21:33 h.


< ifdef >
Pergunta id: 414788, total de respostas: 3
Tempo para primeira resposta: 0:7 h.
Tempo para ultima resposta: 1 Dias e 21:33 h.

Pergunta id: 762833, total de respostas: 3
Tempo para primeira resposta: 0:16 h.
Tempo para ultima resposta: 0:20 h.

Pergunta id: 920560, total de respostas: 17
Tempo para primeira resposta: 0:4 h.
Tempo para ultima resposta maior ou igual a: 7 Anos

< ifndef >
Pergunta id: 722932, total de respostas: 4
Tempo para primeira resposta: 0:3 h.
Tempo para ultima resposta: 0:7 h.

Pergunta id: 1421485, total de respostas: 2
Tempo para primeira resposta: 0:3 h.
Tempo para ultima resposta: 1:45 h.

Pergunta id: 2012496, total de respostas: 8
Tempo para primeira resposta: 0:21 h.
Tempo para ultima resposta: 12 Dias e 21:33 h.
```

Three groups were identified in the set of 1962 questions, as shown in Table 5.17. Note that the total of questions with one or more answers is, therefore, 1658.

It was defined in the context of this work that to analyze the duration of the discussions in the Stack Overflow community, more than one answer to the question is needed. Therefore, the third set of questions was used. The total of these questions and their answers, detailed by words, is shown in the first two columns of Table 5.18.

The Table 5.18 presents the delay averages for the first answer to the questions in days and in hours (both approximate), as well as the general averages. The columns are not complementary, days and hours are equivalent and were used in the display to make it easier to understand. At first glance these times may seem very long, but what happens

Table 5.17 Total of Answers

Group	**Total**
1 – Questions with no answer	304
2 – Questions with one answer	892
3 – Questions with two or more answers	766
Total of Questions	1962

Table 5.18 Total of Questions and Answer and time delay for the first question

keyword	**Question**	**Answer**	**Avrg Days**	**Avrg Hours**
Commonality	81	272	21	493
Variability	69	197	5	124
Decision Model	0	0	0	0
Feature Interaction	0	0	0	0
Feature Dependencies	5	13	29	699
Feature Model	8	16	0	8
FOP (Feat. Orient. Prog.)	1	3	3	71
Product Line	63	192	0	3
Product Family	17	47	0	3
Variation Point	0	0	0	0
Conditional Compilation	334	1055	6	140
ifdef	105	330	24	580
ifndef	83	245	10	230
Total	766	2386		
Total avrg			8.6	206.4

is that some answers can take a great length of time, over a year, and this distorts the value of the average.

Mamykina et al. (2011) also analyzed data from question's answers and reported that they had high averages. Therefore, they focused their analysis on the median of the time series collected for the first answer, which is 11 minutes. In addition, they found that 50% of all questions received the first answer in up to approximately 12 minutes. The data of this work were extracted in the year of 2010 and all the questions and answers contained in the database of Stack Overflow were used until the moment of the research. So we resort to the same kind of analysis she did.

Figures 5.11 and 5.12, respectively, show a box plot and a scattering chart for the delay times for the first answers. The median of the datapoint series of the difference between the time a question was created and its first answer has a median of 13 minutes. It can be observed in both graphs that there is a great concentration of answers with times near the median. In addition, there is a relatively small number of times a little higher and a small number of outliers with values much higher than the median value.

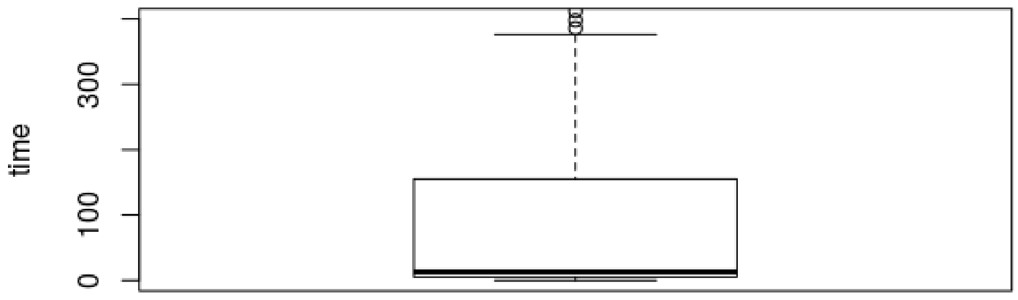


Figure 5.11 Box plot: Time delay for the first answer

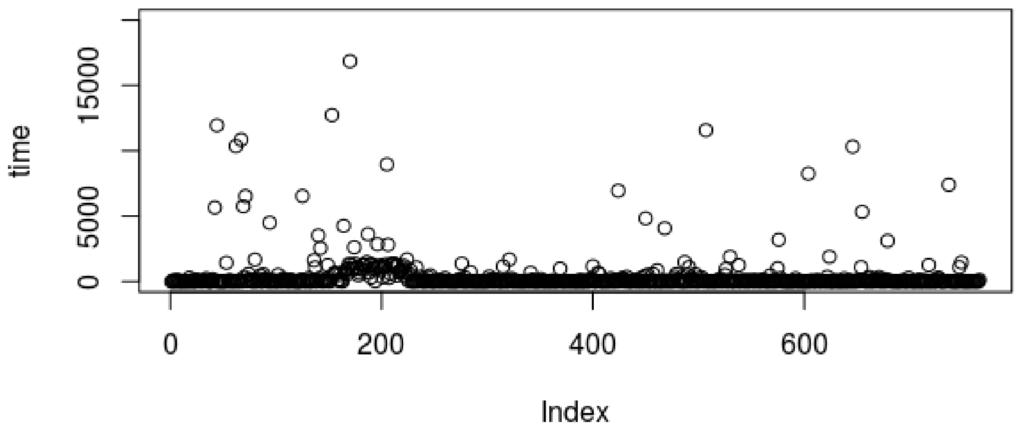


Figure 5.12 Dispersion: Time delay for the first answer

Of the total of 1658 questions with at least one answer, 764 had the first answer in less than 25 minutes, 110 with time between 25 and 90 minutes, 248 with times between 90 and 180 minutes, remaining 536 questions with answers exceeding 180 minutes.

Therefore, it is possible to state that the median obtained in this study is in accordance with the results of Mamykina et al. (2011). Considering the total of 766 questions, 465 had the first answer in less than 25 minutes, 39 with time between 25 and 90 minutes, 77 with times between 90 and 180 minutes, leaving 185 questions with answers that exceed 180 minutes. These data can be observed in Table 5.21.

Table 5.19 Time delay for the last answer for questions with more than one answer

keyword	**Question**	**Avrg Days**	**Avrg Hours**
Commonality	81	795	19084
Variability	69	490	11760
Decision Model	0	0	0
Feature Interaction	0	0	0
Feature Dependencies	5	58	1393
Feature Model	8	12	279
FOP (Feat. Orient. Prog.)	1	7	168
Product Line	63	573	13741
Product Family	17	299	7165
Variation Point	0	0	0
Conditional Compilation	334	940	22549
ifdef	105	496	11327
ifndef	83	219	5250
Total	766	3.889	140.296

The average delay time in Table 5.19 is shown for the last question answer in days and hours, i.e. the time the community took for the final answers. Of these, 148 answers were given in less than 25 minutes, 45 between 25 and 90 minutes, 112 between 90 and 180 minutes and 461 answers were given with times greater than 180 minutes.

The last answers follow the same pattern as the first answers, that is, it was observed that there are questions with a difference ranging from minutes to years. This difference has the approximate value of 461 minutes as a median, which can be seen in the box plot and in the scattering chart shown in figures 5.13 and 5.14, respectively . These figures show a large number of datapoints around the median and also a relatively large number of datapoints at the lower boundary of the upper whisker as well as outliers.

Continuing the analysis, the averages of the difference between the first and last answers, i.e., the time interval between them, in days and hours, are shown in Table 5.20. This time is important because it corresponds to the average discussion duration of these questions after the first answer occurred.

Considering that the total time of a discussion is defined as the difference between the time from the first answer to the last answer, we can then understand that, on average,

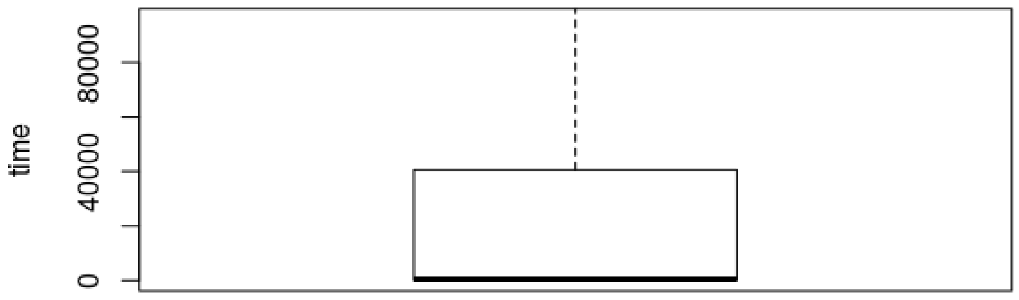


Figure 5.13 Box plot: Time delay for the last answer

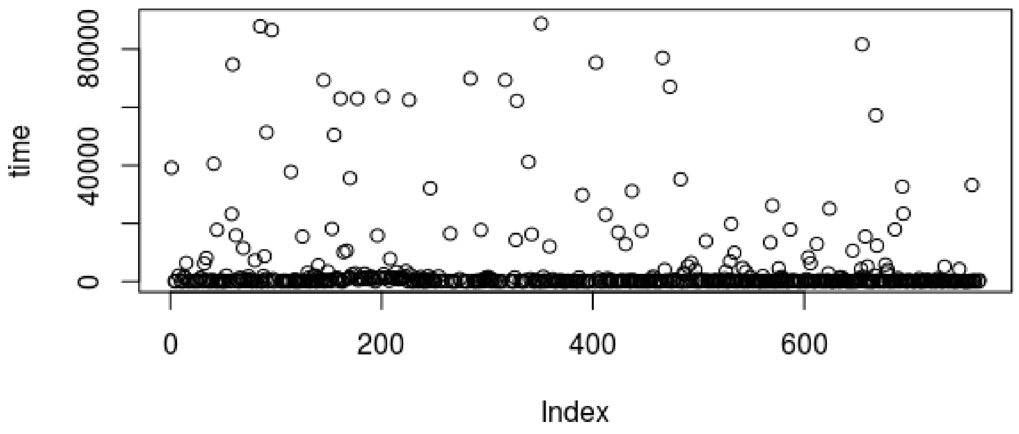


Figure 5.14 Dispersion: Time delay for the last answer

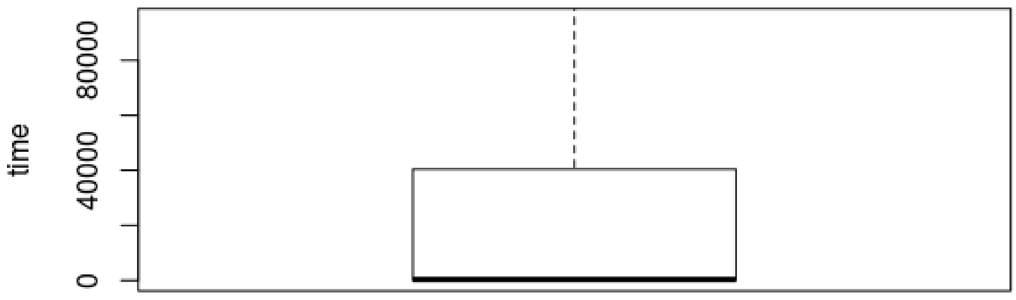


Figure 5.15 Box Plot: Time of discussion

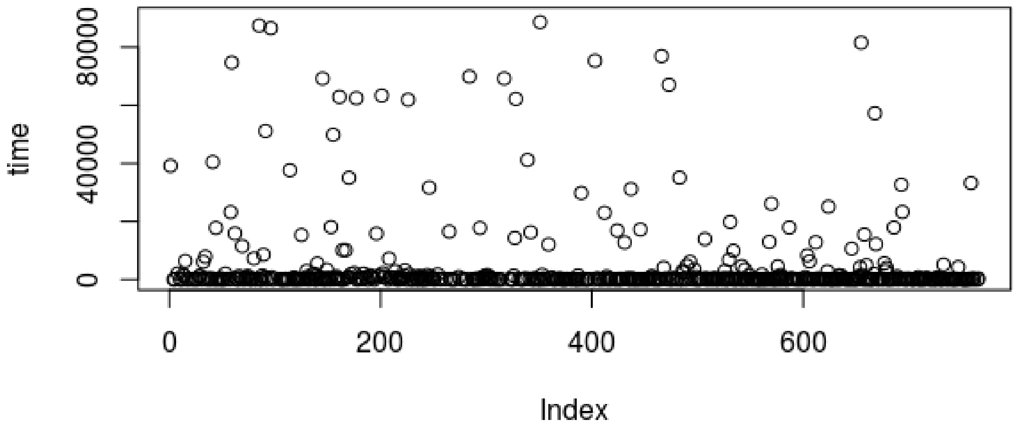


Figure 5.16 Dispersion: Time of discussion

Table 5.20 Time interval between the first and last answers

keyword	Question	Avrg Days	Avrg Hours
Commonality	81	775	18591
Variability	69	485	11637
Decision Model	0	0	0
Feature Interaction	0	0	0
Feature Dependencies	5	29	694
Feature Model	8	11	271
FOP (Feat. Orient. Prog.)	1	4	97
Product Line	63	559	13413
Product Family	17	298	7161
Variation Point	0	0	0
Conditional Compilation	334	934	22409
ifdef	105	472	11327
ifndef	83	209	5014
Total	766	3.784	90616
Total Avrg		291.1	6970

a discussion related to implementation of variability in the community of Stack overflow presents as average value about 291 days.

However, considering the data point series and the median, as shown in the box plot and scattering charts of figures 5.15 and 5.16, the median of the series is 337.5. A large number of outliers with values above the upper whisker limit is also observed.

Considering the set of 766 discussions, 216 lasted less than 25 minutes, 42 had a duration between 25 and 90 minutes and 97 lasted between 90 and 180 minutes. There were 411 discussions lasting more than 180 minutes

Table 5.21 total of questions per time

Time	First	Last	T.Discs.	W.Answer
time $\leq$ 25	465	148	216	764
25 $<$ time $\leq$ 90	39	45	42	110
90 $<$ time $\leq$ 180	77	112	97	248
time $>$ 180	185	461	411	536

The Table 5.22 displays the percentages of results grouped by time limit. We can see that among the first answers given by the community 61% are answered in less than 25 minutes. It is also observed that the delay for the last answer has time greater than 180 minutes in little more than 60% of the cases.

The duration of the discussions follows the values for the last answers, with approximately 54 % of the discussions exceeding the time of 180 minutes. It is also noted that in a time less than 25 minutes approximately 29 % of discussions are concluded (T.Discs.).

The percentage values representing the total of the 1658 questions with at least one answer are In the "W.Answer" column . It can be noted that the values in this column and the values of the "First" column are very close. It is also noted that the central values of columns representing times greater than 25 minutes and less than 180 minutes have very close values.

Table 5.22 Percentages of delay times

Time	**First**	**Last**	**T.Discs.**	**W.Answer**
time $\leq$ 25	60.70	19.32	28.20	46.08
25 $<$ time $\leq$ 90	5.09	5.87	5.48	6.63
90 $<$ time $\leq$ 180	10.05	14.62	12.66	14.96
time $>$ 180	24.15	60.18	53.66	32.33

Continuing with the observations of 1962 questions, with 1658 of them getting at least one answer, we found that 764 of them got their first answer in 25 minutes or less. Of the total questions, 84.51% of them had answers and of those 46.08% had their answers in less than 25 minutes.

We can then conclude that most of the questions are answered within 2 hours, the first answers are fast and occur in about 13 minutes, few questions are answered between 25 minutes and 1:30 minutes. We also observe a good amount of questions that remain unanswered or that their answers have a delay of hours, days and sometimes years; so the answer time is distorted.

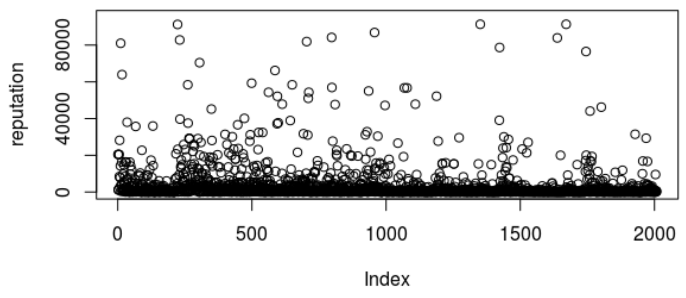
5.4.2 RQ7 - Can we rely on the stack overflow crowd?

There is no direct measure of trust available that can be used to answer this question, to the best of our knowledge. The indirect measure that is commonly used in online communities and also by some researchers to indicate trust is reputation, as discussed in Section 2.4 (GOLBECK, 2005) (JøSANG; GOLBECK, 2009) (RESNICK et al., 2000). Therefore, this is the measure used to answer RQ7.

Table 5.23 User Reputation

	Asker	**Answerer**
Total of users with one question/answer	1.836	2.801
Total of users with more than one question/answer	156	278
Highest reputation value	518.669	1.012.323
Lowest reputation value	1	1
Median reputation	542,5	8.021,5
Reputation average	283	362

From the 1962 questions retrieved and their answers, Stack Overflow users who did them have been identified. Two groups of users were initially created: those who asked the questions and those who answered the questions, as can be seen in Table 5.23. Then, another group of users was created that did not get answers to their questions. It should be noted that the year of collection of the dataset is 2017 and users who asked and answered after that date were not considered.



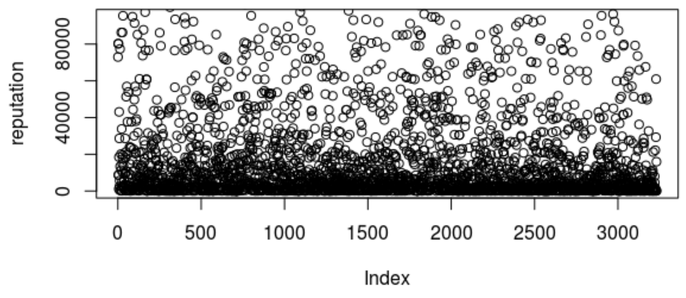


Figure 5.17 Askers (top) and Answerers (below)

The averages and medians shown in Table 5.23 allow us to consider that, in general, the askers are less experienced than the answerers. This conclusion is also reinforced by the dispersion diagrams shown in Figure 5.17. These graphs show that there is a much larger set of answerers with high reputations.

The Table 5.24, on the other hand, shows segmentation of queries and answers for the intervals 0 to 49, 50 to 199, 200 to 1999, 2000 to 4999 and above 5,000. The last line shows a subset of the last segment, of users with a reputation above 20,000.

With the results of users by rank of reputation we constructed the diagram 5.18 where we can observe that for the values below 2,000 the number of the askers is bigger than the numbers of answerers. When it goes beyond that limit the quantity of answering users increased significantly in relation to those who asked. Even considering that the number of answerers is greater than the number of askers, the diagram shows that there are a large number of experienced answerers and that proportionally the number of experienced askers is much lower.

Finally, the two boxplots shown in the figure 5.19 allows to see the difference between the reputations of the users who asked and the users who answered. It should be noted

Table 5.24 Total of users per reputation range

Reputation range	Asker	Answerer
reputation < 50	431	130
reputation >= 50 AND reputation < 200	281	184
reputation >= 200 AND reputation < 2000	549	590
reputation >= 2000 AND reputation <= 5000	256	430
reputation > 5000	319	1.467
reputation > 20000	103	774

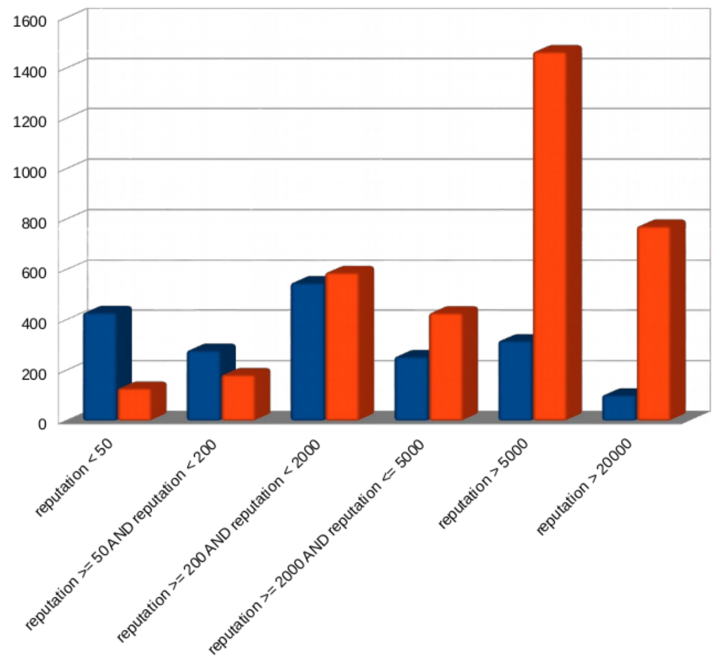


Figure 5.18 Bar diagram - number of users versus Askers (blue) and answerers (red)

that there was a cut in reputations with a value above 20,000 due to restrictions of the R system, which uses exponential notation for intervals above that value, making it difficult to read. This diagram shows the large number of answerer datapoints with a higher reputation than the askers.

Stack overflow provides a list of its users' reputation on its site [1]. Some stand out, as is the case of Jon Skeet, whose profile is shown in figure 5.20. He answered eleven questions selected by our research and is the user with the highest reputation among the selected 1962 respondents as well as the entire Stack Overflow community.

Another highlight is Jonathan Leffler, whose profile is shown in Figure 5.21. He is the most reputable user among those who asked questions among those selected by our study. He asked a question and answered four questions. Figure 5.22 shows an example

[1]https://stackexchange.com/leagues/1/alltime/stackoverflow

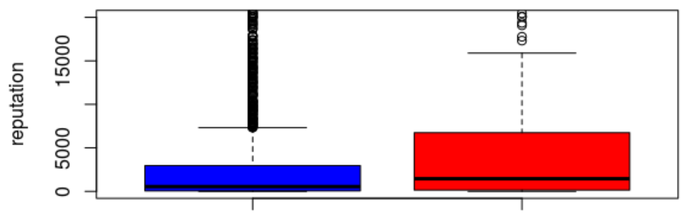
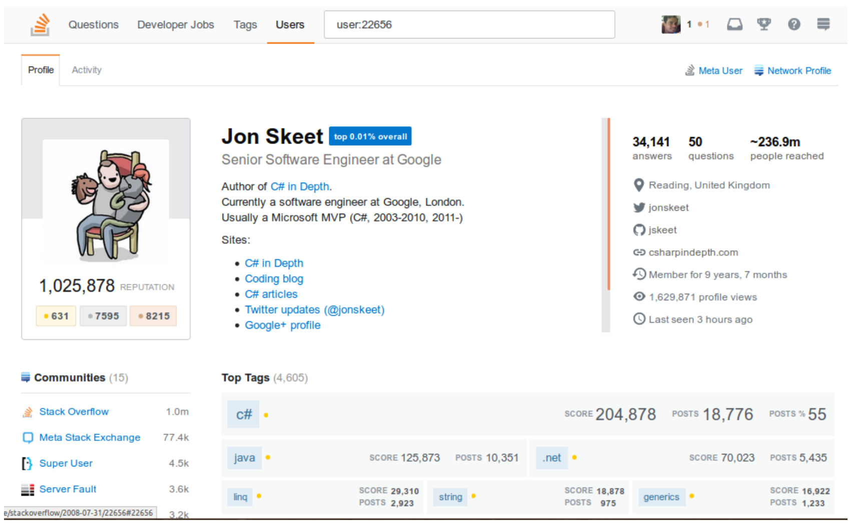


Figure 5.19 Box plot - Reputation of Askers (blue) and Answerers (red)



Jon Skeet top 0.01% overall

Senior Software Engineer at Google

Author of C# in Depth.
Currently a software engineer at Google, London.
Usually a Microsoft MVP (C#, 2003-2010, 2011-)

Sites:

- C# in Depth
- Coding blog
- C# articles
- Twitter updates (@jonskeet)
- Google+ profile

1,025,878 REPUTATION

• 631 • 7595 • 8215

34,141 answers 50 questions ~236.9m people reached

Reading, United Kingdom
jonskeet
jskeet
csharpindepth.com
Member for 9 years, 7 months
1,629,871 profile views
Last seen 3 hours ago

Communities (15)

Stack Overflow	1.0m
Meta Stack Exchange	77.4k
Super User	4.5k
Server Fault	3.6k
	3.2k

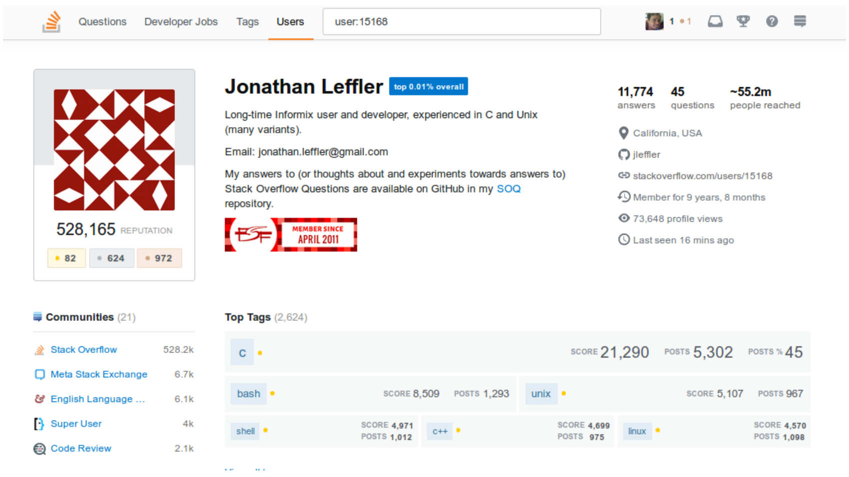
e/stackoverflow/2008-07-31/22656#22656

Top Tags (4,605)

c#	SCORE 204,878	POSTS 18,776	POSTS % 55
java	SCORE 125,873	POSTS 10,351	
.net	SCORE 70,023	POSTS 5,435	
linq	SCORE 29,310	POSTS 2,923	
string	SCORE 18,878	POSTS 975	
generics	SCORE 16,922	POSTS 1,233	

Figure 5.20 User with the greatest reputation among answerers

of answer given by him to a question related to conditional compilation. It is clear that the question refers to a problem involving variability and the code example it shows refers to a common base code (basecase.c) and two features that are selected based on USE_VARIANT_A and USE_VARIANT_C.



Jonathan Leffler

Long-time Informix user and developer, experienced in C and Unix (many variants).

Email: jonathan.leffler@gmail.com

My answers to (or thoughts about and experiments towards answers to) Stack Overflow Questions are available on GitHub in my SOQ repository.

528,165 REPUTATION

● 82 ● 624 ● 972

11,774 answers 45 questions ~55.2m people reached

California, USA

jleffler

stackoverflow.com/users/15168

Member for 9 years, 8 months

73,648 profile views

Last seen 16 mins ago

Communities (21)

Stack Overflow	528.2k
Meta Stack Exchange	6.7k
English Language ...	6.1k
Super User	4k
Code Review	2.1k

Top Tags (2,624)

c	SCORE 21,290	POSTS 5,302	POSTS % 45
bash	SCORE 8,509	POSTS 1,293	
unix	SCORE 5,107	POSTS 967	
shell	SCORE 4,971	POSTS 1,012	
c++	SCORE 4,699	POSTS 975	
linux	SCORE 4,570	POSTS 1,098	

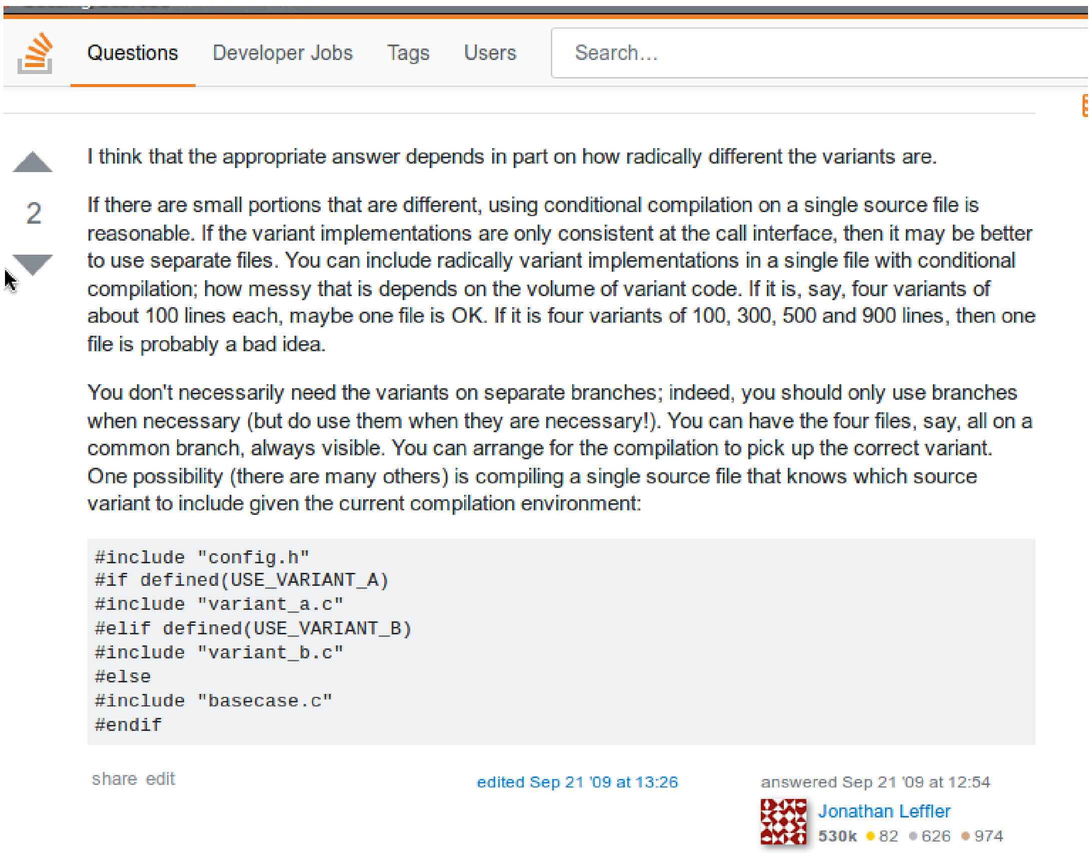
Figure 5.21 User with the greatest reputation among the askers and answerers

Table 5.25 User Reputation

	Unanswered Asker	Asker and Answerer
Total of users	244	239
Highest reputation value	56.700	518.669
Lowest reputation value	1	1
Median reputation	112	1.456
Average reputation	231,42	2.170,16

The Table 5.25 was constructed to study the group of users who had unanswered questions and the group of users who asked and also answered questions. It can be observed that the total users of both groups is very close, but the reputation values of those who did not have their questions answered are much lower in both the average and the median. The segmented data of the two groups are shown in Table 5.26. Note that the vast majority of the 244 askerers in the first group (column 1), 79%, have a reputation below 2,000 points and 95 of them have less than 50 points. In the second group of 239 users found (column 2) the distribution is more homogeneous between groups. Only 34 have a reputation of less than 50 points but, in opposition to that, 105 have a reputation above 2,000 points. In both groups the number of users between 200 and 2,000 reputation points has reputations with close values.

To better study the users who asked, they were divided into three groups: those with no answer, those with only one answer and those with two or more answers. The box plot of these three groups is shown in Figure 5.23, identified in yellow, red and blue,



I think that the appropriate answer depends in part on how radically different the variants are.

If there are small portions that are different, using conditional compilation on a single source file is reasonable. If the variant implementations are only consistent at the call interface, then it may be better to use separate files. You can include radically variant implementations in a single file with conditional compilation; how messy that is depends on the volume of variant code. If it is, say, four variants of about 100 lines each, maybe one file is OK. If it is four variants of 100, 300, 500 and 900 lines, then one file is probably a bad idea.

You don't necessarily need the variants on separate branches; indeed, you should only use branches when necessary (but do use them when they are necessary!). You can have the four files, say, all on a common branch, always visible. You can arrange for the compilation to pick up the correct variant. One possibility (there are many others) is compiling a single source file that knows which source variant to include given the current compilation environment:

```
#include "config.h"
#if defined(USE_VARIANT_A)
#include "variant_a.c"
#elif defined(USE_VARIANT_B)
#include "variant_b.c"
#else
#include "basecase.c"
#endif
```

share edit   edited Sep 21 '09 at 13:26   answered Sep 21 '09 at 12:54
Jonathan Leffler
530k ●82 ●626 ●974

Figure 5.22 Example of answer

Table 5.26 Total of Answered Askers and Askers and Answerere per reputation range

Reputation range	**Unanswered**	**Asker and**
	Asker	**Answerer**
reputation < 50	95	34
reputation >= 50 AND reputation < 200	49	34
reputation >= 200 AND reputation < 2000	61	66
reputation >= 2000 AND reputation <= 5000	23	38
reputation > 5000	16	67
reputation > 20000	4	23

respectively. It is possible to observe in this diagram that the group of users who did not have their questions answered has the reputation value in general inferior to the two groups that received answers and that the group with two or more answers has in general more reputation value. This result is interesting and shows that people with a higher reputation are more likely to get answers. One possible explanation for this is that they ask questions better and have more contacts with other users in the community, they also often and actively participate in the community.

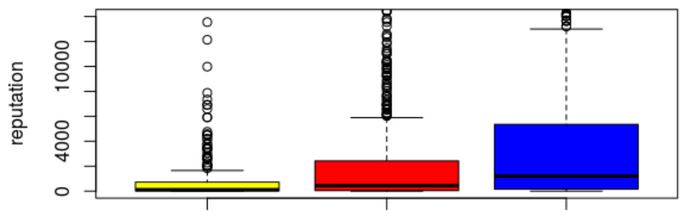


Figure 5.23 Reputation of askers with no answer (yellow), one answer (red) and two or more answers (blue)

5.5 DISCUSSION AND THREATS TO VALIDITY

5.5.1 Discussion

In the context of questions 1 and 2, the analysis of the results produced by LDA showed evidence that the Stack Overflow community has discussed the implementation of variabilities because a large part of the initial set of words was found in the selected question and were grouped into 10 topics more representative. In the RQ1 answer (questions with answers), it was noted that the set of topics was well distributed and that they comprised two main groups of topics: one focused on variability implementation techniques and other composed of more general concepts in the area. In relation to RQ2 (questions without answers), LDA also found topics with bases in the initial vocabulary, but with lower TE and with groups of words smaller and less cohesive, showing that the questions that did not obtain answers, probably were not as well formulated as those that received answers. Being able to detect such situations may be relevant to recommend discussion topics that can be used by experts and project leaders.

With the results obtained in RQ3, we can observe that implementation of variabilities does not have a large number of discussions in the community of Stack Overflow, but the community is attentive to the implementation of variability and, even though it is a small community, it is very active. One of our observations is that some users seem to be unaware of the variability implementation techniques, but perceive problems and opportunities of software reuse problems, also involving software product lines. We also observed a growing trend in research on the subject of research both by the registered community and by web users who search the site.

We used RQ4 to identify which search terms were found in discussions made by the Stack Overflow community. We were able to identify that the terms related to programming were those that received most attention from the community and therefore were the most discussed.

RQ5 led as to focus our efforts to find terms related to implementation of variability that were not discussed in the Stack overflow community. We have identified that most of the terms not discussed are well known in academia and, from the data collected, we gathered that the main reason is that they are not directly related to coding.

Continuing the study in RQ6 we found the result of the delay time for the questions to be answered. Most of them take around 2 hours but in many cases it takes about 13 minutes to get the first answer. We also found questions that remain unanswered or that their answers take years to appear, and this then distorts the result of the average answer time.

We used RQ7 to analyze users of the Stack Overflow community who discussed the implementation of variability. We used the user reputation indicator defined by the Stack Overflow site to evaluate the reliability of the questions and answers selected in the search. As a result, we found that higher-ranking users are more successful in their questions as they are better elaborated and thus more likely to receive answers, as well as having answers accepted among the ones they have given to question of other users. That is possible because they are very active in the community. We concluded that we can trust these users.

5.5.2 Threats to validity

The relatively small number of questions retrieved when compared to the total size of the dataset is a threat to the results obtained by this work. Even so, it is necessary to take into account that the number of 1962 datapoints is relevant for empirical research in the area of software engineering.

An action mitigates this problem. Whenever possible, we compare the indicators obtained with other indicators published by researchers and by Stack Overflow site itself, such as the average number of answers by questions and the average time delay of answers. The results obtained were always comparable with the general set of questions and with other studies.

A second problem is the lack of confirmation of the relative knowledge about implementation of variability by Stack Overflow users. Even without this confirmation, we can accept the knowledge of these users, understanding that they represent a heterogeneous and comprehensive set of IT professionals. In addition, the questions and answers analyzed in this dissertation indicates that questioners and answerers have knowledge of the subject.

Finally, in this dissertation, the collaborating specialists are not identified. This is not the purpose of this work.

Chapter

6

CONCLUSION

6.1 FINAL CONSIDERATIONS

An empirical investigation was carried out in this dissertation, trying to understand how the theme "implementation of variability" is discussed by the community of Information Technology professionals who use the Stack Overflow site. Our research interest was formulated with seven research questions (RQ).

Initially, a list of 52 words related to this theme was defined based on the literature of the area and the opinion of experts. They were then used as the initial vocabulary for the algorithm that implements the LDA technique in order to find the main topics related to the variability implementation, as well as to validate the words in the list. This was done in the analysis of research questions 1 and 2.

Then, with the most used words in the questions and answers, as validated by the LDA algorithm, searches were performed on the dataset retrieved from the Stack Overflow site and several studies were carried out to find the best set to be analyzed, considering that some words used alone recovered many questions, because they have other uses in the IT area. The best result was obtained by combining these words two by two, and thus 1962 questions were retrieved.

These 1962 were then analyzed in research questions 3 to 7 which focused on the following questions: discover the mechanisms of implementation of variabilities more discussed, less discussed and not discussed; what is the delay in responding to the formal questions on implementation of variabilities and how long the discussions last; and finally, what is the profile of Stack Overflow users who ask and answer questions, as well as the degree of confidence we can have about the quality of the questions and answers presented.

6.2 MAIN CONTRIBUTIONS

The main contributions and results of this research are as follows:

- Definition of a list of words that refer to variability implementation concepts and techniques that are most commonly used by the Stack Overflow developer community.

- Several research results (or findings) on the set of questions – and answers – selected, among which we highlight the following:
  - Two groups of words were identified as those most used by the Stack Overflow community: one more conceptual and one more related to variability implementation techniques.
  - The average delay time for the first answer to the questions and the duration of the discussion when there is more than one answer.
  - The study on the degree of confidence in the Stack overflow user group that asked questions or answered questions based on the reputation indicator of community members.

6.3 FUTURE WORKS

New research works can be carried out in the future to continue this work and explore some points raised here but not deepened. One of them would be to use the list of words most used by the community of Stack Overflow on implementation of variabilities as the basis for other studies, using other forums or even to guide discussions about the subject.

Another interesting research would be to make an analysis of the selected questions and answers related to the implementation of variabilities for feeling detection in software engineering.

The third proposal of new studies would be to carry out a study to identify specialists in software reusability among the users of Stack overflow. In addition, a more qualitative exploratory study could be carried out with the specialists found to confirm the results found or to seek other information about this community and how it deals with reuse of software and, more specifically, with the implementation of variabilities.

BIBLIOGRAPHY

ALLAMANIS, M.; SUTTON, C. Why, when, and what: Analyzing stack overflow questions by topic, type, and code. In: *Proceedings of the 10th Working Conference on Mining Software Repositories*. Piscataway, NJ, USA: IEEE Press, 2013. (MSR '13), p. 53–56. ISBN 978-1-4673-2936-1. Disponível em: ⟨http://dl.acm.org/citation.cfm?id=2487085.2487098⟩.

BABAR, M. A.; CHEN, L.; SHULL, F. Managing variability in software product lines. *IEEE Software*, v. 27, n. 3, p. 89–91, 94, May 2010. ISSN 0740-7459.

BAJAJ, K.; PATTABIRAMAN, K.; MESBAH, A. Mining questions asked by web developers. In: *Proceedings of the 11th Working Conference on Mining Software Repositories*. [S.l.: s.n.], 2014. p. 112–21.

BARUA, A.; THOMAS, S. W.; HASSAN, A. E. What are developers talking about? An analysis of topics and trends in stack overflow. *Empirical Software Engineering*, v. 19, n. 3, p. 619–654, Jun 2014. ISSN 1573-7616. Disponível em: ⟨https://doi.org/10.1007/s10664-012-9231-y⟩.

BLEI, D. M.; NG, A. Y.; JORDAN, M. I. Latent dirichlet allocation. *J. Mach. Learn. Res.*, JMLR.org, v. 3, p. 993–1022, mar. 2003. ISSN 1532-4435. Disponível em: ⟨http://dl.acm.org/citation.cfm?id=944919.944937⟩.

BOSCH, J.; CAPILLA, R.; HILLIARD, R. Trends in systems and software variability [guest editors' introduction]. *IEEE Software*, v. 32, n. 3, p. 44–51, May 2015. ISSN 0740-7459.

CALEFATO, F. et al. Mining successful answers in stack overflow. In: *Proceedings of the 12th Working Conference on Mining Software Repositories*. Piscataway, NJ, USA: IEEE Press, 2015. (MSR '15), p. 430–433. ISBN 978-0-7695-5594-2. Disponível em: ⟨http://dl.acm.org/citation.cfm?id=2820518.2820579⟩.

CHEN, C.; XING, Z. Mining technology landscape from stack overflow. In: *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA: ACM, 2016. (ESEM '16), p. 14:1–14:10. ISBN 978-1-4503-4427-2. Disponível em: ⟨http://doi.acm.org/10.1145/2961111.2962588⟩.

CHEN, L.; BABAR, M. A. A systematic review of evaluation of variability management approaches in software product lines. *Information and Software Technology*, v. 53, n. 4, p. 344 – 362, 2011. ISSN 0950-5849. Special section: Software Engineering track of the 24th Annual Symposium on Applied Computing. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0950584910002223⟩.

CLEMENTS, P.; BACHMANN, F. *Variability in Software Product Lines*. Carnegie Mellon University, Pittsburgh, Pennsylvania,, 2005.

GACEK, C.; ANASTASOPOULES, M. Implementing product line variabilities. In: *Proceedings of the 2001 Symposium on Software Reusability: Putting Software Reuse in Context*. New York, NY, USA: ACM, 2001. (SSR '01), p. 109–117. ISBN 1-58113-358-8. Disponível em: ⟨http://doi.acm.org/10.1145/375212.375269⟩.

GOLBECK, J. A. *Computing and Applying Trust in Web-based Social Networks*. Tese (Doutorado) — University of Maryland, 2005.

HOFFMAN, M. D.; BLEI, D. M.; BACH, F. Online learning for latent dirichlet allocation. In: *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*. USA: Curran Associates Inc., 2010. (NIPS'10), p. 856–864. Disponível em: ⟨http://dl.acm.org/citation.cfm?id=2997189.2997285⟩.

HOFFMAN, M. D. et al. Stochastic variational inference. *J. Mach. Learn. Res.*, JMLR.org, v. 14, n. 1, p. 1303–1347, maio 2013. ISSN 1532-4435. Disponível em: ⟨http://dl.acm.org/citation.cfm?id=2567709.2502622⟩.

HONG, L.; DAVISON, B. D. Empirical study of topic modeling in twitter. In: *Proceedings of the First Workshop on Social Media Analytics*. New York, NY, USA: ACM, 2010. (SOMA '10), p. 80–88. ISBN 978-1-4503-0217-3. Disponível em: ⟨http://doi.acm.org/10.1145/1964858.1964870⟩.

JøSANG, A.; GOLBECK, J. Challenges for robust trust and reputation systems. 10 2009.

KAVALER, D. et al. Using and asking: APIs used in the android market and asked about in stackoverflow. In: JATOWT, A. et al. (Ed.). *Social Informatics*. Cham: Springer International Publishing, 2013. p. 405–418. ISBN 978-3-319-03260-3.

LINARES-VáSQUEZ, M.; DIT, B.; POSHYVANYK, D. An exploratory analysis of mobile development issues using stack overflow. In: *2013 10th Working Conference on Mining Software Repositories (MSR)*. [S.l.: s.n.], 2013. p. 93–96. ISSN 2160-1852.

MAMYKINA, L. et al. Design lessons from the fastest q&a; a site in the west. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2011. (CHI '11), p. 2857–2866. ISBN 978-1-4503-0228-9. Disponível em: ⟨http://doi.acm.org/10.1145/1978942.1979366⟩.

PANICHELLA, A. et al. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In: *Proceedings of the 2013 International Conference on Software Engineering*. Piscataway, NJ, USA: IEEE Press, 2013. (ICSE '13), p. 522–531. ISBN 978-1-4673-3076-3. Disponível em: ⟨http://dl.acm.org/citation.cfm?id=2486788.2486857⟩.

PARNIN, C.; TREUDE, C.; GRAMMEL, L. *Crowd documentation: Exploring the coverage and the dynamics of API discussions on stack overflow.* 2012. Georgia Institute of Technology Technical Report GIT-CS-12-05.

RESNICK, P. et al. Reputation systems. *Commun. ACM*, ACM, New York, NY, USA, v. 43, n. 12, p. 45–48, dez. 2000. ISSN 0001-0782. Disponível em: ⟨http://doi.acm.org/10.1145/355112.355122⟩.

SVAHNBERG, M.; GURP, J. van; BOSCH, J. A taxonomy of variability realization techniques: Research articles. *Softw. Pract. Exper.*, John Wiley & Sons, Inc., New York, NY, USA, v. 35, n. 8, p. 705–754, jul. 2005. ISSN 0038-0644. Disponível em: ⟨http://dx.doi.org/10.1002/spe.v35:8⟩.

THOMAS, S. W. et al. Modeling the evolution of topics in source code histories. In: *Proceedings of the 8th Working Conference on Mining Software Repositories.* New York, NY, USA: ACM, 2011. (MSR '11), p. 173–182. ISBN 978-1-4503-0574-7. Disponível em: ⟨http://doi.acm.org/10.1145/1985441.1985467⟩.

TREUDE, C.; BARZILAY, O.; STOREY, M.-A. How do programmers ask and answer questions on the web? (nier track). In: *Proceedings of the 33rd International Conference on Software Engineering.* New York, NY, USA: ACM, 2011. (ICSE '11), p. 804–807. ISBN 978-1-4503-0445-0. Disponível em: ⟨http://doi.acm.org/10.1145/1985793.1985907⟩.

VALE, T. et al. Software product lines traceability: A systematic mapping study. *Information and Software Technology*, v. 84, p. 1 – 18, 2017. ISSN 0950-5849. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0950584916304463⟩.

VENKATESH, P. K. et al. What do client developers concern when using web apis? an empirical study on developer forums and stack overflow. In: *2016 IEEE International Conference on Web Services (ICWS).* [S.l.: s.n.], 2016. p. 131–138.

VILLELA, K. et al. A survey on software variability management approaches. In: *Proceedings of the 18th International Software Product Line Conference - Volume 1.* New York, NY, USA: ACM, 2014. (SPLC '14), p. 147–156. ISBN 978-1-4503-2740-4. Disponível em: ⟨http://doi.acm.org/10.1145/2648511.2648527⟩.

ZOU, J. et al. Towards comprehending the non-functional requirements through developers eyes. *Inf. Softw. Technol.*, Butterworth-Heinemann, Newton, MA, USA, v. 84, n. C, p. 19–32, abr. 2017. ISSN 0950-5849. Disponível em: ⟨https://doi.org/10.1016/j.infsof.2016.12.003⟩.

Appendix A

AN EXAMPLE OF A DATASET REGISTER

```
<row Id="36" PostTypeId="1" AcceptedAnswerId="352" CreationDate=
"2008-08-01T12:35:56.917" Score="100" ViewCount="43086" Body=
"&lt;p&gt;How can I monitor an SQL Server database for changes
to a table without using triggers or modifying the structure of
the database in any way? My preferred programming environment is
&lt;ahref=&quot;http://en.wikipedia.org/wiki/.NET_Framework&quot;
rel=&quot;noreferrer&quot;&gt;.NET&lt;/a&gt; and C#.&lt;/p&gt;&#x
A;&#xA;&lt;p&gt;I'd like to be able to support any &lt;ahref=&quot;
http://en.wikipedia.org/wiki/Microsoft_SQL_Server#Genesis&quotrel=
&quot;noreferrer&quot;&gt;SQL Server 2000&lt;/a&gt; SP4 or
newer. My application is a bolt-on data visualization for another
company's product. Our customer base is in the thousands, so I
don't want to have to put in requirements that we modify the
third-party vendor's table at every installation.&lt;/p&gt;&#xA;
&#xA;&lt;p&gt;By&lt;em&gt;&quot;changes to a table&quot;&lt;/em&gt;
I mean changes to table data, not changes to table structure.&lt;/
p&gt;&#xA;&#xA;&lt;p&gt;Ultimately, I would like the change to
trigger an event in my application, instead of having to check
for changes at an interval.&lt;/p&gt;&#xA;&#xA;&lt;hr&gt;&#xA;&#xA;
&lt;p&gt;The best course of action given my requirements (no triggers
or schema modification, SQL Server 2000 and 2005) seems to be
to use the &lt;code&gt;BINARY_CHECKSUM&lt;/code&gt; function in &lt;a
href=&quot;http://en.wikipedia.org/wiki/Transact-SQL&quot;
rel=&quot;noreferrer&quot;&gt;T-SQL&lt;/a&gt;. The way I plan to
implement is this:&lt;/p&gt;&#xA;&#xA;&lt;p&gt;Every X seconds run
the following query:&lt;/p&gt;&#xA;&#xA;&lt;pre&gt;&lt;code&gt;SELECT
CHECKSUM_AGG(BINARY_CHECKSUM(*))&#xA;FROM sample_table&#xA;WITH
(NOLOCK);&#xA;&lt;/code&gt;&lt;/pre&gt;&#xA;&#xA;&lt;p&gt;And compare
that against the stored value. If the value has changed, go through
```

```
the table row by row using the query:&lt;/p&gt;&#xA;&#xA;&lt;pre&gt;
&lt;code&gt;SELECT row_id, BINARY_CHECKSUM(*)&#xA;FROM sample_table
&#xA;WITH (NOLOCK);&#xA;&lt;/code&gt;&lt;/pre&gt;&#xA;&#xA;&lt;p&gt;
And compare the returned checksums against stored values.&lt;/p&gt;
&#xA;" OwnerUserId="32"LastEditorUserId="2571493"LastEditorDisplayName=
"Mark Harrison"LastEditDate="2016-11-30T07:50:37.517"LastActivityDate=
"2017-02-27T13:35:29.970"Title="Check for changes to an SQL Server
table?"Tags="&lt;sql&gt;&lt;sql-server&gt;&lt;datatable&gt;&lt;rdbms&gt;"
AnswerCount="8"CommentCount="1" FavoriteCount="21" />
```

Appendix
B

REPUTATION RULES IN STACK OVERFLOW

```
A reputação é uma medição aproximada do grau de confiança da
comunidade em você; ela é obtida convencendo seus colegas de
que você sabe do que está falando. O uso básico do site,
inclusive fazer perguntas, responder e sugerir edições, não
exige reputação alguma. Mas quanto mais reputação você
obtiver, mais privilégios ganhará. A principal forma de ganhar
reputação é publicando boas perguntas e respostas úteis. Os
votos nessas publicações farão com que você ganhe (ou às vezes
perca) reputação. Observe que os votos em publicações
marcadas como "wiki da comunidade" não geram reputação alguma.
Você pode ganhar no máximo 200 pontos de reputação por dia de
qualquer combinação das atividades a seguir. Apenas as
gratificações recebidas e as respostas aceitas não estão
sujeitas ao limite diário de reputação.

Você ganha reputação quando:
  a pergunta recebe votos a favor: +5
  a resposta recebe votos a favor: +10
  a resposta é marcada como \aceita": +15 (+2 para quem aceitou)
  uma edição sugerida é aceita: +2 (até um total de +1000 por usuário)
  sua resposta recebeu uma gratificação: +quantidade total
    da gratificação
  uma de suas respostas recebeu uma gratificação automaticamente:
     +0.5 da quantidade da gratificação
  bônus de associação ao site: +100 em cada site (recebida no
       máximo uma vez por site)

Se for um usuário experiente da rede Stack Exchange com 200 ou
mais pontos de reputação em pelo menos um site, você receberá
```

um bônus inicial de reputação de +100 pontos para superar as restrições básicas de novo usuário. Isso acontecerá automaticamente em todos os sites atuais do Stack Exchange onde você tiver uma conta e em qualquer outro site Stack Exchange no momento que fizer login.

Você perde reputação quando:
sua pergunta recebe votos contra: 2
sua resposta recebe votos contra: 2
você vota contra uma resposta: 1
você coloca uma gratificação em uma pergunta: quantidade total da gratificação
uma de suas publicações recebe 6 sinalizações como spam ou ofensiva:100

Todos os usuários começam com um ponto de reputação e a reputação nunca pode cair abaixo de 1. A aceitação de sua própria resposta não acrescenta pontos de reputação. Se um usuário reverter um voto, o ganho ou perda de reputação correspondente será revertido também. A reversão do voto como resultado de uma fraude de votação também retornará a reputação perdida ou ganha.

No fim deste espectro de reputação há pouca diferença entre os usuários com alta reputação e os moderadores. Isso é intencional. Não administramos este site. Ele é administrado pela comunidade.

Appendix C

COMBINATION OF WORDS

"commonality" and "variant"
"commonality" and "variant"
"commonality" and "aggregation"
"commonality" and "parameterization"
"commonality" and "reflection"
"commonality" and "decision model"
"commonality" and "feature interaction"
"commonality" and "feature dependencies"
"commonality" and "feature model"
"commonality" and "product line"
"commonality" and "product family"
"commonality" and "variation point"
"commonality" and "conditional compilation"
"commonality" and "design pattern"
"commonality" and "ifdef"
"commonality" and "ifndef"

"variability" and "aggregation"
"variability" and "parameterization"
"variability" and "reflection"
"variability" and "decision model"
"variability" and "feature interaction"
"variability" and "feature dependencies"
"variability" and "feature model"
"variability" and "product line"
"variability" and "product family"
"variability" and "variation point"
"variability" and "conditional compilation"
"variability" and "design pattern"

"variability" and "ifdef"
"variability" and "ifndef"

"variability" and "variant"
"variability" and "aggregation"
"variability" and "parameterization"
"variability" and "reflection"
"variability" and "decision model"
"variability" and "feature interaction"
"variability" and "feature dependencies"
"variability" and "feature model"
"variability" and "product line"
"variability" and "product family"
"variability" and "variation point"
"variability" and "conditional compilation"
"variability" and "design pattern"
"variability" and "ifdef"
"variability" and "ifndef"

"variant" and "aggregation"
"variant" and "delegation"
"variant" and "parameterization"
"variant" and "reflection"
"variant" and "decision model"
"variant" and "feature interaction"
"variant" and "feature dependencies"
"variant" and "feature model"
"variant" and "product line"
"variant" and "product family"
"variant" and "variation point"
"variant" and "conditional compilation"
"variant" and "design pattern"
"variant" and "ifdef"
"variant" and "ifndef"

"aggregation" and "delegation"
"aggregation" and "parameterization"
"aggregation" and "reflection"
"aggregation t" and "decision model"
"aggregation" and "feature interaction"
"aggregation" and "feature dependencies"
"aggregation" and "feature model"
"aggregation" and "product line"
"aggregation" and "product family"

"aggregation" and "variation point"
"aggregation" and "conditional compilation"
"aggregation" and "design pattern"
"aggregation" and "ifdef"
"aggregation" and "ifndef"

"delegation" and "parameterization"
"delegation" and "reflection"
"delegation" and "decision model"
"delegation" and "feature interaction"
"delegation" and "feature dependencies"
"delegation" and "feature model"
"delegation" and "product line"
"delegation" and "product family"
"delegation" and "variation point"
"delegation" and "conditional compilation"
"delegation" and "design pattern"
"delegation" and "ifdef"
"delegation" and "ifndef"

"parameterization" and "reflection"
"parameterization" and "decision model"
"parameterization" and "feature interaction"
"parameterization" and "feature dependencies"
"parameterization" and "feature model"
"parameterization" and "product line"
"parameterization" and "product family"
"parameterization" and "variation point"
"parameterization" and "conditional compilation"
"parameterization" and "design pattern"
"parameterization" and "ifdef"
"parameterization" and "ifndef"

"reflection" and "decision model"
"reflection" and "feature interaction"
"reflection" and "feature dependencies"
"reflection" and "feature model"
"reflection" and "product line"
"reflection" and "product family"
"reflection" and "variation point"
"reflection" and "conditional compilation"
"reflection" and "design pattern"
"reflection" and "ifdef"
"reflection" and "ifndef"

"decision model" and "feature interaction"
"decision model" and "feature dependencies"
"decision model" and "feature model"
"decision model" and "product line"
"decision model" and "product family"
"decision model" and "variation point"
"decision model" and "conditional compilation"
"decision model" and "design pattern"
"decision model" and "ifdef"
"decision model" and "ifndef"

"feature dependencies" and "feature model"
"feature dependencies" and "product line"
"feature dependencies" and "product family"
"feature dependencies" and "variation point"
"feature dependencies" and "conditional compilation"
"feature dependencies" and "design pattern"
"feature dependencies" and "ifdef"
"feature dependencies" and "ifndef"

"feature model" and "product line"
"feature model" and "product family"
"feature model" and "variation point"
"feature model" and "conditional compilation"
"feature model" and "design pattern"
"feature model" and "ifdef"
"feature model" and "ifndef"

"product line" and "product family"
"product line" and "variation point"
"product line" and "conditional compilation"
"product line" and "design pattern"
"product line" and "ifdef"
"product line " and "ifndef"

"product family" and "variation point"
"product family" and "conditional compilation"
"product family" and "design pattern"
"product family" and "ifdef"
"product family" and "ifndef"

"variation point" and "conditional compilation"
"variation point" and "design pattern"

"variation point" and "ifdef"
"variation point" and "conditional compilation"

"conditional compilation" and "design pattern"
"conditional compilation" and "ifdef"
"conditional compilation" and "ifndef"

"design pattern" and "ifdef"
"design pattern" and "ifndef"

"ifdef" and "ifndef"

Appendix D

KEYWORDS FOR VARIABILITY IMPLEMENTATION

Table D.1 List of keywords

Keyword	Keyword	Keyword
aggregation	ahead	aspectorientedprogramming
aspectweaving	bindingtime	collectionofvariants
commonality	composition	conditionalcompilation
conditiononconstant	conditiononvariable	configuration
constraintprogramming	decisionmodel	delegation
derivation	designfeatures	dynamicclassloading
dynamiclinklibraries	dynamicvariability	featuredependencies
featureide	featureinteraction	featuremodel
featuremodeling	foda	fop
frames	framework	functionalityforbinding
ifdef	ifndef	inheritance
instatiation	interface	makefile
modularfeature	operationaldependencies	overloading
parameterization	productfamily	productline
reflection	reuse	staticlibraries
variability	variabilityinspace	variabilitymanagement
variabilityscope	variant	variantfeature
variationpoint		

Appendix E

LIST OF STOP WORDS

install, err, path,implemented, compilers, book, problem, display, eof, book, original, scripts, -, keyword, target, listed, top, mvc, days, created, logging, sheet, standard, free, linear, rm, rc, network, xp, extract, series, debug, grammar, xe, array, arraysize, hidden, writing, enter, answer, example, root, visible, widget, net, selected, errors, studio, upload, cmd, updated, void, matlab, verdana, calculate, limited, ive, gt, lt, a, able, about, above, abst, accordance, according, accordingly, across, act, actually, added, adj, affected, affecting, affects, after, afterwards, again, against, ah, all, almost, alone, along, already, also, although, always, am, among, amongst, an, and, announce, another, any, anybody, anyhow, anymore, anyone, anything, anyway, anyways, anywhere, apparently, approximately, are, aren, arent, arise, around, as, aside, ask, asking, at, auth, available, away, awfully, b, back, be, became, because, become, becomes, becoming, been, before, beforehand, begin, beginning, beginnings, begins, behind, being, believe, below, beside, besides, between, beyond, biol, both, brief, briefly, but, by, c, ca, came, can, cannot, can't, cause, causes, certain, certainly, co, com, come, comes, contain, containing, contains, could, couldnt, d, date, did, didn't, different, do, does, doesn't, doing, done, don't, down, downwards, due, during, e, each, ed, edu, effect, eg, eight, eighty, either, else, elsewhere, end, ending, enough, especially, et, et-al, etc, even, ever, every, everybody, everyone, everything, everywhere, ex, except, f, far, few, ff, fifth, first, five, fix, followed, following, follows, for, former, formerly, forth, found, four, from, further, furthermore, g, gave, get, gets, getting, give, given, gives, giving, go, goes, gone, got, gotten, h, had, happens, hardly, has, hasn't, have, haven't, having, he, hed, hence, her, here, hereafter, hereby, herein, heres, hereupon, hers, herself, hes, hi, hid, him, himself, his, hither, home, how, howbeit, however, hundred, i, id, ie, if, i'll, im, immediate, immediately, importance, important, in, inc, indeed, index, information, instead, into, invention, inward, is, isn't, it, itd, it'll, its, itself, i've, j, just, k, keep, keeps, kept, kg, km, know, known, knows, l, largely, last, lately, later, latter, latterly, least, less, lest, let, lets, like, liked, likely, line, little, 'll, look, looking, looks, ltd, m, made, mainly, make, makes, many, may, maybe, me, mean, means, meantime, meanwhile, merely, mg, might, million, miss, ml, more, moreover, most, mostly, mr, mrs, much, mug, must, my, myself, n, na, name, namely, nay, nd, near, nearly, necessarily, necessary, need, needs, neither, never, nevertheless, new, next, nine, ninety, no, nobody, non, none, nonetheless, noone, nor, normally, nos, not, noted, nothing, now, nowhere, o, obtain, obtained, obviously, of, off, often, oh, ok, okay, old, omitted, on, once, one, ones, only, onto, or, ord, other, others, otherwise, ought, our, ours, ourselves, out, outside, over, overall, owing, own, p, page, pages, part, particular, particularly, past, per, perhaps, placed, please, plus, poorly, possible, possibly, potentially, pp, predominantly, present, previously, primarily, probably, promptly, proud, provides, put, q, que, quickly, quite, qv, r, ran, rather, rd, re, readily, really, recent, recently, ref, refs, regarding, regardless, regards, related, relatively, research, respectively, resulted, resulting, results, right, run, s, said, same, saw, say, saying, says, sec, section, see, seeing, seem, seemed, seeming, seems, seen, self, selves, sent, seven, several, shall, she, shed, she'll, shes, should, shouldn't, show, showed, shown, showns, shows, significant, significantly, similar, similarly, since, six, slightly, so, some, somebody, somehow, someone, somethan, something, sometime, sometimes, somewhat, somewhere, soon, sorry, specifically, specified, specify, specifying, still, stop, strongly, sub, substantially, successfully, such, sufficiently, suggest, sup, sure, than, that, that's, the, their, theirs, them, themselves, then, there, there's, these, they, they'd, they'll, they're, they've, this, those, through, to, too, under, until, up, very, was, wasn't, we, we'd, we'll, we're, we've, were, weren't, what, what's, when, when's, where, where's, which, while, who, who's, whom, why, why's, with, won't, would, wouldn't, you, you'd, you'll, you're, you've, your, yours, yourself, yourselves