Extracting and Modeling Product Line Functional Requirements

Nan Niu Steve Easterbrook
Department of Computer Science, University of Toronto
Toronto, Ontario, Canada M5S 3G4
{nn,sme}@cs.toronto.edu

Abstract

We introduce an extractive approach to building a product line's requirements assets. We define the functional requirements profiles (FRPs) according to the linguistic characterization of a domain's action-oriented concerns, and show that FRPs can be extracted from a document based on domain-aware lexical affinities that bear a 'verb – direct object' relation. The validated FRPs are then amenable to semantic case analysis so as to uncover the variation structures. Finally, merging FRPs helps discover the requirements interdependencies. We use orthogonal variability modeling to represent the product line's external variability and constraints. We apply our approach to an auto-marker product line. The study shows our approach complements domain analysis by quickly offering insights into system functionalities and product line variabilities.

1 Introduction

A software product line (SPL) succeeds because mass customization is achieved by variability management, i.e., by exploiting the family members' commonalities and by controlling their differences [12, 32]. Requirements assets enhance the effectiveness of reuse as developers can work on the abstractions closer to the SPL's initial concepts.

Many contemporary SPL methods, such as FODA [22] and FAST [36], base requirements definition on heavyweight domain analysis, and so do many requirements engineering (RE) techniques for SPLs like PRS [14] and definition hierarchies [25]. In practice, the up-front cost and the level of manual effort associated with domain analysis present a prohibitive adoption barrier for many organizations that could otherwise benefit.

To ease the transition from a single-system mentality to software mass customization, Krueger proposed the extractive adoption model as a means of reusing existing products for the SPL's initial baseline [24]. Core assets are no longer created from scratch, but are mined from software repositories. The extractive approach is particularly effective for an organization that has accumulated development experiences and artifacts in a domain and wants to quickly transition from conventional to SPL engineering. Notably, the main beneficiaries are small and medium-sized enterprises (SMEs) as large companies tend to proactively launch a SPL in the mature market segment [9]. The basic tenets of the extractive model are:

- Maximal reuse: A fair amount of grounded knowledge is embedded in legacy systems, and a SPL's assets can be established on top of existing artifacts.
- Gradual change: Incrementally exposing small variations avoids over-specifying and inaccurately predicting a complete set of SPL features.
- Reactive development: Each under-specified asset will be enriched when abstractions are refactored as they emerge from an evolving SPL.

Leveraging this adoption model urges us to consider fundamental questions like where and what to extract, and how to represent the result. In this paper, we propose a semi-automated approach to identifying functional requirements assets by analyzing natural language (NL) documents. Our aim is to reduce the manual operation cost and increase the operation efficiency in domain analysis.

Studies of RE practice in SMEs, such as [16] and [6], showed that the majority of requirements are written in NL because text is used universally to convey information and to communicate. We therefore choose NL documents to be the primary extraction source, and anticipate our textual-based technique can effect a wide spectrum of domains.

We adopt the orthogonal variability model (OVM) proposed by Pohl et al. [32] to represent the extraction result. An OVM defines a SPL's variability in a single view, so we can consistently manage the variability across requirements, design, realization, and testing artifacts. The building blocks of the OVM are variation points, variants, dependencies, and constraints. These elements provide a conceptual basis for determining what should be extracted.

When constructing a SPL's requirements assets, we shall follow two principles [32]: 1) Focus more on *external* variability (visible to customers) and less on *internal* variability (useful to implementers), 2) Focus more on *what*

varies (variation point) and less on *how* it varies (variants). Our strategy is to tease out functional requirements profiles (FRPs) and analyze their variabilities. We define the notion of FRP to capture the domain's action themes and to define a context for studying system qualities [30]. The FRPs in each document are identified on the basis of lexical affinities [27] and "verb – direct object" relations [34]. We then use Fillmore's case theory [15] to characterize each FRP's semantics. Note that Liaskos et al. used Fillmore's cases to acquire variability for goal models [26]; here, we deal with textual requirements. Merging FRPs allows us to discover the variability dependencies and constraints, thereby forming an initial OVM for the SPL.

The contributions of our work lie in the concept of FRP and the support for extracting and modeling FRPs for a SPL. Our approach complements existing domain analysis methods by quickly offering insights into system functionalities and their variabilities, and the approach is readily scalable and extensible. To mitigate the risk of being overgeneral, domain concepts are incorporated when possible. We study an auto-marker SPL to demonstrate our approach. The results are promising and comparable to expert opinions. In addition, we present several scenarios for using the extracted FRPs to show the benefits of our approach.

2 Motivating Example

We motivate our work with a scenario for extracting requirements assets in an auto-marker SPL. The system employed in the University of Toronto for marking programming assignments exhibited delays in turnaround time, due to the administrative burden of printing, distributing assignments to teaching assistants (TAs), and returning assignments to students. To provide feedback to students in a timely manner, a dozen teams, each consisting of 3 to 4 junior undergraduates, conducted requirements analysis and wrote software requirements specifications (SRS's) for Web-based auto-markers in their course projects [1].

Note that these team projects were carried out separately by interviewing different sets of stakeholders. The resulting SRS's shared certain concerns, but also had different focuses. Consolidating these results could help understand the commonalities and variations in the auto-marker domain. All 12 auto-marker SRS's followed the IEEE-STD-830 standard in a textual form [19]. Figures 1a and 1b show the excerpts from two SRS's in the repository.

We are interested in culling a set of functional requirements profiles (FRPs) from these SRS's. We define FRPs to be the action-oriented concerns [34] that bear a high information value of a document [27]. FRPs model the user-visible system functionalities, and are represented by "verb – direct object" pairs. Figure 1d shows a partial list of FRPs extracted from the auto-marker SRS's.

SPL engineering considers it crucial to define a set of standard terms used in discussions about and descriptions

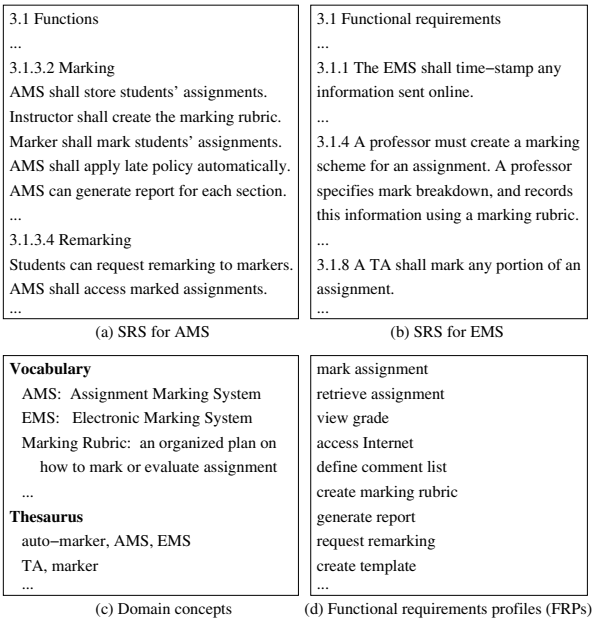

Figure 1. FRPs extraction example

of the domain, and makes developing a domain dictionary part of the core assets [36]. Figure 1c depicts a snippet of the auto-marker domain concepts: Thesaurus identifies synonym classes [33], whereas vocabulary provides the definitions of terms, acronyms, and abbreviations required to properly interpret the requirements documents [19]. These concepts are identified by domain experts. According to Figure 1c, we would treat "marking rubric" as a single conceptual unit, and thus determine the FRP "create marking rubric", as indicated in Figure 1d.

Although the extracted FRPs are capable of characterizing the SPL's action themes, the flat list (Figure 1d) hinders us from gaining insights into the variability structures and dependencies. We analyze the FRP's variabilities by filling out Fillmore's semantic cases [15]: agentive, objective, process, etc. Figure 2a shows two sample case structures: each case defines a variation dimension for the FRP, and a case's values determine the range of that dimension. For example, only a "TA" can "mark assignment", and the types of assignment to be marked can be "late" or "on-time".

We take advantage of the OVM notations [32] to rigorously express the SPL's variability. Figure 2b illustrates an OVM, in which a "VP" triangle represents a variation point (what can vary) and a "V" box represents a variant (how the "VP" varies). We map each FRP to a "VP" and organize the variants by the FRP's cases. A mandatory variant is linked by a solid line, whereas optionals are linked by dotted lines. The alternative choice among the optionals is further annotated with an arch, along with the cardinalities specified in $[min..max]$. The variability constraint, such as "FRP_1 requires FRP_4", is given by an arrow in Figure 2b.

We focus on the FRPs in building the requirements as-

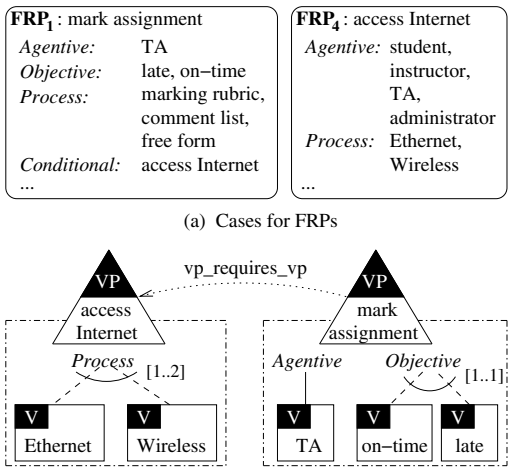

Figure 2. Semantic cases and OVM

sets for a couple of reasons. First, a FRP represents the functional aspect of a feature, which is an essential characteristic of an application domain [22]. Features are distinctively identifiable abstractions that must be implemented, tested, delivered, and maintained [21]. While system qualities, such as reusability and sustainability, may become the prominent features in the long run, product functionalities remain the salient features directly observable by users.

Second, as pointed out by Bosch, starting from the functional requirements does not preclude the optimization of quality requirements during the architectural design stages [7]. In fact, we found out that having a set of concrete FRPs could effectively align quality requirements [30]. While the role of FRP in SPL engineering is further explored in Section 5, we now consider how to extract the FRPs from an existing requirements document.

3 Functional Requirements Extraction

The central question that we address in this section is that: Given a NL document, how can its characterizing attributes, which relate to system functionalities, be produced? We examine some single-term indexing schemes, and propose a technique for automatically generating the FRPs. We use the auto-marker SRS's to evaluate the effectiveness and cost of our approach.

3.1 Single-Term Indexing

When constructing the indices for a requirements artifact, information retrieval (IR) techniques draw information from the texts rather than from a human expert. Instead of relying on a great deal of manually pre-encoded semantic information to create a knowledge base, little semantic knowledge is required and no interpretation of the document is given in most IR techniques [33]. Automatic indexing systems attempt to characterize the document rather than understand it. We prefer IR techniques in our work for reasons of cost, scalability, and domain transportability [27].

An important issue in indexing is the nature of the indices. The most usual form is a single-term index, where a term is a content identifier typically encapsulated in a word. The assumption underlying the single-term indices is stated in Zipf's law: Given some corpus of NL utterances, the frequency of any word is inversely proportional to its rank in the frequency table [33]. John proposed some basic techniques for integrating legacy documentation assets into a SPL [20]. Use cases and system functionalities can be identified by searching the verbs with the highest frequencies of occurrence within a document. We can greatly improve the results by filtering out a list of stop words (e.g., "be" and "have") that commonly appear in all domains [20]. The left column of Table 1 shows the top-ranked verbs in the AMS SRS by frequency of occurrence.

Although term frequency indicates relevance, some noise exists, mainly due to words appearing too often in a given context. In order to reduce the influence of such words it is necessary to identify the most representative terms, i.e., those containing the most information. The quantity of information of a word within a corpus is defined by its Shannon information content as:

$$\mathrm{INFO}(w) = -\log_2(P\{w\}) \quad (1)$$

where $P\{w\}$ is the observed probability of occurrence w in the corpus [33]. Therefore the more frequent a word is in a domain, the less information it carries. The middle column of Table 1 ranks the AMS SRS's verbs by $\mathrm{INFO}(w)$. Most term-ranking strategies, such as tf-idf and signal-noise ratio [33], take $\mathrm{INFO}(w)$ into account. Also note that the information value defined in (1) lies at the heart of many single-term IR applications in RE (e.g., [11], [18]).

3.2 Functional Requirements Profiles (FRPs)

Extracting valuable conceptual information from documentation can be done by using richer indexing units than single words. Maarek et al. used a two-word unit, called lexical affinity (LA), for profiling software libraries [27]. In linguistics, an LA between two units of language stands for a correlation of their common appearance in the utterances of the language. An LA is more restrictive than a simple co-occurrence since it necessarily relates words that are involved in a modifier-modified relation. LAs in large textual corpora have been shown to convey information on both syntactic and semantic levels, and to provide us with a powerful way of taking context into account [27].

For our purposes, we restrict the definition of LAs by observing them within a finite requirements document rather than within the whole language so as to retrieve *conceptual* affinities rather than purely *lexical* ones. One limitation of considering only a two-word unit as an LA is that domain concepts are not preserved. For example, "marking rubric"

Input: a requirements document d, a domain vocabulary voc
Output: a list of LAs (and frequencies of occurrence) from d concerning voc
Preprocessing
  For each entry e in voc
    Replace every occurrence of e in d by u_e
  For each word w in d **AND** $w \notin u_e$
    $u_w \leftarrow$ inflectional root of w
Main Procedure
  Initialize Hashtable la_freq
  For each $u \in (u_e \cup u_w)$ from the beginning to the end of d
    Let $u_1, \ldots, u_m$ be the m units immediately following u in d
    (where $m = 5$ except the end of sentence is reached earlier)
    For $i = 1$ to m
      $f \leftarrow la\_freq.getValue(\{u, u_i\})$
      ($f = 0$ when $\{u, u_i\}$ has not been encountered before)
      $la\_freq.put(\{u, u_i\}, f + 1)$
  Return Hashtable la_freq

Figure 3. Extracting domain-aware LAs

would be treated as two separate words, not as one proper term, in [27]. To address this problem, we augment our approach with a semantic component, as shown in Figure 1c, so that each entry in the domain vocabulary, voc, is maintained as one atomic conceptual unit.

Figure 3 outlines the algorithm for extracting domain-aware LAs. Two steps are involved in preprocessing. First, every domain concept appearing in the document is replaced by a single unit. In our implementation, for example, "marking rubric" (e), defined by the domain experts, is replaced by "marking_rubric" (u_e). Second, the remaining words are stemmed [13] to permit an accurate identification of the LAs.[1] As a result, the word is represented by its inflectional root, i.e., the singular form for nouns and the infinitive form for verbs. The order of these two preprocessing steps is important; otherwise, "marking rubric" would be stemmed into "mark rubric", causing the domain concept to be unrecognizable in the document.

The main procedure of the algorithm is built upon the sliding window technique [27]. The idea is to make use of an empirical observation that 98% of lexical relations relate words which are separated by at most 5 words within a single sentence [28]. Therefore, most of the LAs involving a conceptual unit u can be extracted by examining the neighborhood of each occurrence of u within a span of 5 units. The window is slid throughout the document d without crossing sentence boundaries. It is worth bearing in mind that the window size of 5 conceptual units is not a parameter but a property of the English language [28]. Given that both the window size and the domain vocabulary entries are bounded by some small constants, the extraction of domain-aware LAs is linear in the number of conceptual units in the document. In the worst case where voc is not defined, the complexity of the algorithm in Figure 3 is $\mathcal{O}(n)$, where n is the number of words in d.

Similar to single-term indexing, using frequency to directly determine relevance may introduce some noise. From the definition in (1), we infer the definition of the *quantity of information* (ρ) of an LA in a given document d as:

$$\begin{aligned} \rho((\{u_1, u_2\}, f)) &= f \times \text{INFO}(\{u_1, u_2\}) \\ &= f \times -\log_2(P\{u_1, u_2\}) \\ &\approx f \times -\log_2(P\{u_1\} \times P\{u_2\}) \end{aligned} \quad (2)$$

where $(\{u_1, u_2\}, f)$ is a tuple retrieved while analyzing d, meaning $\{u_1, u_2\}$ is an LA appearing f times in d, as defined by the hashtable la_freq in Figure 3. To simplify the computation of $\text{INFO}(\{u_1, u_2\})$, we consider u_1 and u_2 as independent variables. This assumption represents only an approximation within the textual universe, as noted in [27].

The ρ score defined in (2) measures the information value an LA carries based on both its frequency of appearance in the text and the quantity of information of the conceptual units involved. The LAs with high ρ scores thus effectively characterize the requirements document, but they typically include several modifier-modified relations. Consider the following sentence, taken from the EMS SRS in Figure 1b:

> "A professor specifies mark breakdown, and records this information using a marking rubric."

Some of the potential LAs in this sentence are:

- of type verb-DO (direct object), e.g., "specify breakdown", "record information";
- of type subject-verb, e.g., "professor specify"; or
- of type noun-noun, e.g., "mark breakdown".

We are concerned only with the verb-DO relation since our goal is to construct functional profiles. Shepherd studied the verb-DO pairs in source code and observed their denotations of action-oriented concerns [34]. More generally, an especially strong relationship exists between verbs and their themes in English. A theme is the subject matter that the action (implied by the verb) acts upon, and usually appears as a DO [10]. Thus, we define the *functional requirements profiles* (FRPs) of a document to be the domain-aware LAs that have a high information value (ρ) and bear a verb-DO relation. The right column of Table 1 lists the FRPs extracted from the AMS SRS, along with the ρ values.

Table 1. Profiling the AMS SRS (3,264 words)

Verbs	Freq.	Verbs	INFO	FRPs	ρ
mark	64	accept	11.67	mark assignment	106.75
access	18	install	11.67	access assignment	96.31
submit	14	highlight	11.67	divide number	69.85
release	14	define	11.67	notify instructor	61.97
provide	12	calculate	10.67	modify information	54.68
use	9	update	10.67	release assignment	44.58
notify	9	check	10.09	change password	43.10
...	...	...	...	...	...

3.3 Evaluation

It is apparent from Table 1 that FRPs are more appealing indices than single words because each term in the FRP provides a context that helps disambiguate the other. But how much better, or more importantly how effective, are FRPs in

[1] We use OpenNLP [5] to perform stemming, and discuss the evaluation in Section 3.3.

characterizing the user-visible system functionalities? What is the effort required to generate the FRPs? What counts as a high ρ value? We answer these questions empirically by analyzing the SRS's in the auto-marker repository.

We fully implemented the single-term indexing schemes and the FRP-extraction procedure. We used OpenNLP [5] for stemming and part-of-speech (POS) tagging. State-of-the-art taggers like OpenNLP have the precision of about 97%, which makes them unlikely to become an extra error source [23]. Nevertheless, we believe FRPs can tolerate certain tagging and stemming errors because phrasal indices offer a robust way of taking context into account; testing this hypothesis is beyond the scope of our current work.

For single-term evaluations, documents were indexed using stemmed verbs,[2] ranked by raw term frequencies in Verb_Freq and by term information values in Verb_INFO. To achieve a fair comparison of FRPs, we discarded the LA containing any stop word [3], and then kept only the LAs whose first units were tagged as a verb by OpenNLP [5]. We did not check if an LA's second unit was a DO because the single-term indices did not have such a component. We did not perform synonym substitution either (Figure 1c) because Verb_Freq and Verb_INFO did not consider it. In the future, we plan to implement these operations, and expect to generate more effective FRPs than the ones appeared in the current evaluation. The FRPs were ranked by the ρ scores.

We adopted FAST [36], one of the most mature SPL development methods, as a gold standard in evaluating the indexing schemes. FAST uses practical domain engineering for a SPL's scope, commonality, and variability analysis. We organized three weekly meetings following FAST's detailed process guidelines: the first was to establish standard terminology, the second was to define the SPL's externally perceivable functionalities, and the third was to compare the FAST's domain analysis results with the automatic indexing results. Each meeting lasted about one hour. Three domain experts participated in all these meetings: a TA who also acted as the moderator, an instructor, and a student.

– **Effort.** As aforementioned, the algorithm for extracting domain-aware LAs is linear in the number of words contained in the document, so are the single-term indexing schemes. The cost of further obtaining verbs and FRPs is highly dependent on POS tagging and stemming. In our experience, it took OpenNLP about 5 seconds to process a 5,000-word auto-marker SRS on a PC with a 2GHz P-4 CPU and 768Mb RAM. Recent advances in requirements tagging and stemming (e.g., [13]), together with the computational efficiency of our extraction algorithm, give us confidence that our approach to functional requirements profiling can be readily scalable and extensible.

Analyzing existing requirements documents is not a replacement, but a complement, to FAST. FAST requires a considerable amount of time and effort, which is not uncommon among domain analysis methods. Our goal is to use automatic indexing to help domain experts gain insights into a SPL's functionalities quickly and effectively.

[2] A stop list of common words must be removed from the indices [20, 13]. We used a domain-neutral list [3] in our current implementation.

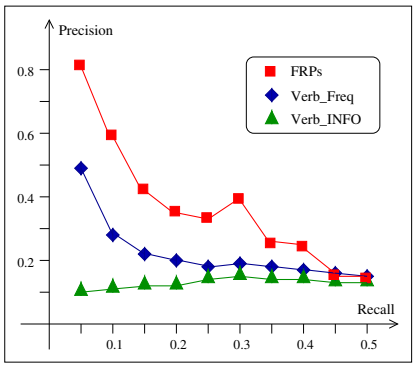

Figure 4. Precision-recall curves

– **Effectiveness.** We assessed the quality of indices via well-known IR metrics, in comparison with the FAST's results and the assessment of domain experts. Our FAST meetings were organized such that the experts tried to define the SPL's functionalities *before* they were asked to evaluate the indices generated automatically. This helped alleviate the bias toward automatic indexing. Relevance judgments of Verb_Freq, Verb_INFO, and FRPs were performed independently by the three domain experts, and different opinions were reconciled in the last FAST meeting.

We use *precision* and *recall* [33] to assess an indexing scheme's effectiveness. Precision measures accuracy and is defined as the proportion of extracted information which is relevant. Recall measures coverage and is defined as the proportion of extracted relevant information to the total amount of all relevant information. The effectiveness comparison was performed by measuring, for the indexing schemes, precision at several levels of recall. We used the 12 auto-marker SRS's and followed the procedure [33]:

1) Plotting precision-recall points for each SRS with each plot corresponding to a given recall value;
2) Extrapolating the plots to obtain precision values for recall values that were not explicitly observed; and
3) Deriving from the curves computed in stage 2) the average precision values at fixed recall intervals to obtain a single curve for the indexing scheme considered.

We have built such curves for Verb_Freq, Verb_INFO, and FRPs. The curves are shown on the same axes in Figure 4, where 10 fixed recall values are plotted for each indexing scheme. The best performance is reached by the scheme whose curve is closest to the area where both precision and recall are maximized – the upper right corner of the graph. The bump of the FRPs curve is due to the inability of 4 SRS's to reach the 30% recall level or beyond; for the remaining 8 SRS's, the average precisions keep decreasing

for the recall values greater than 30%. The $\mathrm{Verb\_Freq}$ curve slightly indicates such a fluctuation. The $\mathrm{Verb\_INFO}$ curve, to our surprise, is so flat that the indices are indifferent. This may suggest $\mathrm{Verb\_INFO}$ should not be applied directly, but it certainly warrants further investigation.

The results in Figure 4 show that for the sample SRS's, the FRPs are better characterizations of system functionalities than the single-term indices. From Figure 4, it is clear that on average, FRPs have 46% better precision than $\mathrm{Verb\_Freq}$, and 181% than $\mathrm{Verb\_INFO}$. This suggests that our extraction results are much more accurate. FRPs therefore can be a good starting point for the stakeholders to understand and discuss the domain.

It turns out that, using the current analysis method, the maximum recall achieved by all three schemes is approximately the same, around 58% on the average. This implies that the extraction result of an individual system/product may not achieve a very high coverage of the domain. On one hand, as we improve our extraction method, we anticipate to achieve higher recall and precision. On the other hand, the current analysis method may be misleading. For example, we may achieve a much higher recall if we use the union, instead of the average, to plot the recall curve at fixed precision intervals. We are currently comparing different analysis methods in IR [33] to assess our results.

– **Threshold.** Although FRPs are promising indices, we are left with a practical problem: which FRPs shall be considered primarily. We address this question based on the ρ score defined in (2), which measures an FRP's information value. Maarek et al. [27] used $\rho \geq \bar{\rho} + \sigma$ as the cutoff value for profiling software libraries, where $\bar{\rho}$ represents the mean and σ the standard deviation of the distribution of ρ within one document. In our experiment, this threshold was so selective that, for every SRS, many relevant FRPs were filtered out. Comparing all 12 SRS's from our dataset indicated that $\rho \geq \bar{\rho}$ was likely to be an optimal threshold, but a more decisive answer to how to select useful FRPs would require further evaluation.

Choosing a threshold helps filter out insignificant FRPs, but the resulting FRPs can still contain irrelevant information or miss relevant information. However, this seeming drawback is really an advantage: Before the FRPs can become a SPL's assets, they must be validated by the domain experts. From our experience with the auto-marker SPL, a $5,000$-word SRS resulted in around 20 FRPs, which typically took an expert less than 10 minutes to validate. Our approach, hence, provides an efficient way to complement FAST and other domain analysis methods.

4 Functional Variability Modeling

4.1 Semantic Cases

The validated FRPs are amenable to semantic analysis so as to uncover their variation structures. We use Fillmore's case theory [15] as a basis for understanding language semantics in an RE context; though, here we focus on functional requirements. The theory analyzes the surface syntactic structure of sentences by studying the combination of *cases* (i.e., semantic roles like agent, object, location, etc.) which are required by a specific verb. According to Fillmore, there exists an essential set of cases that fits in the case system of every known language. Each of these universal cases addresses a particular semantic concern of the verb in a sentence, and each represents a potential semantic slot that may or must be associated with the verb. Hence, given a verb, a *case frame* can be defined, which is a set of semantic slots that the verb *evokes*. As an example, the verb "open"is necessarily associated with an *objective* slot ("*what* opens/is opened?") but may also be associated with an *agentive* slot (to answer "*who* opens?").

FRPs have made the DO role explicit because the verb-DO relation renders the action and its theme in English [10]. The discovery of variation structures can be driven by identifying the essential cases associated with the verb in every FRP. In this context, a case defines a variation dimension, i.e., a question whose alternative answers result in alternative refinements of the original action-oriented concern expressed by the FRP. The collection of all dimensions relevant to an FRP determines the *variation structure*, or the variation frame, evoked by the FRP.

Following Fillmore's idea of defining a universal set of cases, we introduce a general set of dimensions for conceptualizing the FRP's variation structure. The set we considered includes most of the semantic roles Fillmore originally proposed, but also draws information from recent work on variability frames for goals [26]. A high-level goal can be refined by studying the cases associated with the goal's verb. However, a high-level goal in [26] is expressed mostly by a verb-DO pair, such as "send message" or "display record". Such concerns will likely be recognized as FRPs in our approach. Thus, we consider the following variation dimensions for an FRP.

• *Agentive* defines the agent(s) whose activities will bring about the FRP's state of affairs. Responses to this question are typically actors or combinations of actors found in the domain, including the system-to-be. For example, $\{\text{machine, TA, instructor}\}_{Agentive}$ "check time stamp".

• *Objective* defines the object(s) that is affected by the FRP's activity. Since a DO is already part of the FRP, this case concerns mainly with the *types* of DO involved. For example, "mark $\{\text{late, on-time}\}_{Objective}$ assignment".

• *Locational* defines the spatial location(s) where the FRP's activity is supposed to take place. For example, "mark assignment" $\{\text{in the lab, at home}\}_{Locational}$.

• *Temporal* defines the duration or frequency of the FRP's activity. For example, "keep log" for $\{\text{a term, a month, a week}\}_{Temporal}$.

• *Process* refers to the instrument ($P.ins$) used, as well as the means ($P.mea$) and the manner ($P.man$) by which the FRP's activity is performed. Some examples are, "access Internet" via $\{$Ethernet, Wireless$\}_{P.ins}$, "mark assignment" $\{$following marking rubric, in free form$\}_{P.mea}$, or "adjust mark" $\{$dramatically, subtly$\}_{P.man}$.[3]

• *Conditional* defines the trigger(s) of the FRP's action or the condition(s) under which the FRP's function can be achieved. For example, "mark assignment" only if $\{$"access Internet", "retrieve assignment"$\}_{Conditional}$.

The set is by no means an exhaustive list of grammatical features that must be associated with functional requirements descriptions, but a catalogue of categories that can help analysts understand the variation points, i.e., *what* can vary, of the FRP. Two case structures are illustrated in Figure 2a. Our experience showed that systematically identifying the variation point could uncover its variants (*how* it varies) that would otherwise remain hidden. For instance, it was when "mark *late* assignment" was identified that we noticed that *on-time* assignments should be marked as well.

4.2 Variability Constraints

We now discuss the intra- and inter-FRP variability issues [32]. Our purpose is to identify the variability dependencies and constraints so that FRPs can be integrated to form the SPL's initial asset base. To that end, we present several heuristic rules for variability interdependency identification. It is important to keep in mind that variability management requires a deep understanding of the domain. Our heuristics serve as an aid to this understanding and should be treated as such. Our work is guided by the OVM framework [32]. As shown in Figure 5, we extend the OVM by adding a boundary for each FRP to mark its internal variation structure — the semantic cases and their corresponding values. The idea is to allow the user to zoom in (display) or zoom out (hide) the internal structure of any FRP to gain a comprehensive view of the OVM.

The intra-FRP variability refers to the values identified along each of the case dimensions. We concentrate more on the case's mandatory or optional property, and less on the connection between the cases within a single FRP. Note that not every FRP evokes all the cases described earlier. It is up to the domain engineer to decide which intra-FRP variabilities to model. For example, the *locations* where "mark assignment" occurs are not deemed essential for the auto-marker SPL, so Figure 5 disregards the *locational* case for "mark assignment".

> *Heuristic 1*: "If a case is associated with only one value, then the case has one mandatory variant."

This heuristic often applies to the agentive role to indicate the sole actor who shall perform the action. In Figure 5, the agent of "mark assignment" and that of "create marking rubric" instantiate this heuristic.

[3] We will not distinguish $P.ins$, $P.mea$, and $P.man$ for the remainder of the paper, but use the general $Process$ dimension instead.

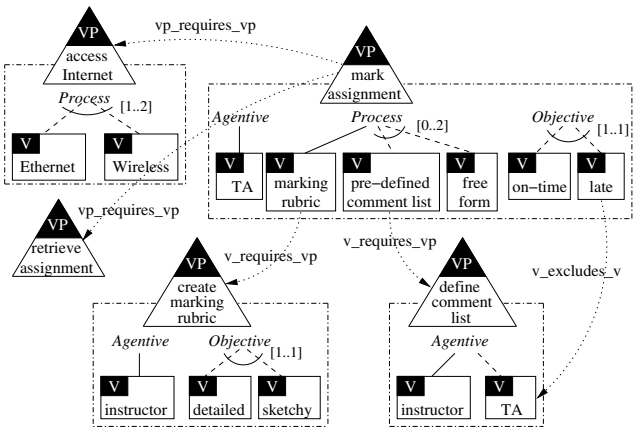

Figure 5. Partial OVM for the auto-marker SPL

> *Heuristic 2*: "If diverse values are identified for a case, then alternative choice(s) should be made."

An example is "keep log" for $\{$a term, a month, a week$\}_{Temporal}$, where the values are diverse but not mutually exclusive. A $[min..max]$ choice among the variants is often made in such situations, but also note that one variant (e.g., a month) can subsume another (e.g., a week). A special case of this heuristic is that the values for a case are opposite or contradictory, as in "mark $\{$late, on-time$\}_{Objective}$ assignment". In such cases, a unique $[1..1]$ choice is made.

The inter-FRP variability constraint refers to the 'requires' or 'excludes' relationship between a variant and a variation point, between two variants, or between two variation points [32]. In Figure 5, such a constraint is represented by an annotated, dotted arrow.

> *Heuristic 3*: "If FRP$_\beta$ is conditional to FRP$_\alpha$, then there exists a vp_requires_vp constraint from FRP$_\alpha$ to FRP$_\beta$."

This heuristic helps identify two constraints in Figure 5: "mark assignment" requires both "access Internet" and "retrieve assignment". The *conditional* semantic case usually reveals FRPs' interdependencies, so it seldom appears as an intra-FRP variation dimension.

> *Heuristic 4*: "If DO$_\delta$ is a case value of FRP$_\phi$, then there exists a v_requires_vp constraint from DO$_\delta$ to some FRP$_\gamma$ such that the direct object of FRP$_\gamma$ is DO$_\delta$."

For example, marking rubric (DO$_\delta$) is a *process* value of "mark assignment" (FRP$_\phi$), so the v_requires_vp constraint exists from marking rubric to "create marking rubric" (FRP$_\gamma$). Figure 5 shows this constraint, as well as a similar one on the DO comment list.

The above heuristic rules stem from our experience of merging the FRPs extracted from different auto-marker

SRS's [1, 8]. They fit our intuition and lead to a trial-and-error inquiry into the SPL's variability dependencies and constraints. Currently, only simple relationships can be identified by applying the heuristics manually. While we are keen to discover more rules and patterns, we insist they should all play a supporting role for expert opinions, because many relationships, such as $[min..max]$ choice, 'excludes' constraint, and conflicts, are not likely to be inferred from text, or even from the FRPs, directly.

We conducted a preliminary evaluation by asking a domain expert, who did not attend the FAST meetings (cf. Section 3.3) and would have fresh eyes, to review the integrated auto-marker OVM. She not only confirmed the OVM's value as an asset, by which the SPL can be managed and evolved as a single and unified entity, but also spotted several constructs and relationships in the OVM to be the *new* insights into the domain: some examples were FRPs "highlight code segment", "define comment list", the *process* variants pre-defined comment list of "mark assignment", and the v_excludes_v constraint from "mark *late* assignment" to the comment list defined by a *TA*.

The response was encouraging in that the OVM did not surprise the domain expert by containing obviously incorrect information. Not only that, the consolidated FRPs and their variabilities helped uncover the incompleteness and inconsistency to a certain degree. The OVM thus demonstrated its usefulness by guiding people in keeping an eye on the important issues. Although the initial feedback from the expert's review was positive, in the longer term, we plan to carry out more in-depth empirical studies to determine the value of our approach.

5 Discussion and Applications

Having detailed how to extract and model a SPL's FRPs, we now consider what they are good for. Before describing the usage scenarios, we discuss two key aspects of our approach.

5.1 Levels of Variability

The variabilities that are identified at each phase of core assets development have different levels of abstraction. FRPs work at a primitive level. According to Moon et al. [29], a primitive requirement (PR) is a transaction that has an effect on an external actor; sample PRs for the online news domain are "write an opinion" and "forward an article by e-mail". We enhance the idea of PR by heeding the conceptual units in the domain corpus that carry high information value and concern action themes. Thus, the primitive-level FRP can be used as a building block of a more complex refinement.

Some extractive approaches like [20] use a naive way to decide the variability attribute: a requirement is mandatory if it intersects the documents being extracted; otherwise it is optional to the SPL. This method violates the underspecify principle of assets mining [24]: any extraction result is intrinsically incomplete. Decisions drawn from incomplete information can be grossly inaccurate. We emphasize that experts must validate the extracted FRPs and domain knowledge shall be the major driver in determining the variability properties at all granularity levels.

5.2 Scalability and Extensibility

Although the auto-marker SPL is relatively small and has modest business goals, it suffices to show the applicability and effectiveness of our approach. Our technique for extracting FRPs is scalable because 1) the algorithm for identifying lexical affinities (cf. Figure 3) is computationally efficient, 2) exploiting available NLP toolset for tagging and stemming does not present a considerable overhead, and 3) the extraction process is *summarizing*, which means the output (FRPs) is significantly smaller than the input (the requirements document) [17]. Experimenting with the large-size SRS's of NASA's family of fault tolerant system services [4] resulted in a compelling summarizing-factor around 200: on average, 101 FRPs were identified for each of the three SRS's whose average size was $20,477$ words.

Domain semantics play a key role in modeling variabilities and determining domain concepts (cf. Figure 1c), thereby affecting the extensibility, i.e., domain transportability, of our approach. Domain knowledge is critical to building sensible assets (e.g., FRPs and OVM), and developing domain-specific heuristics can increase reuse. In terms of domain concepts identification, most RE standards (e.g., [19]) contain a definition section for specifying the domain terminology. We counted on this source in our auto-marker study: each SRS's domain vocabulary had about 10 entries. For the sake of comparing indexing schemes, we did not perform synonym substitution (cf. Section 3.3). Therefore some sensitivity analysis is in order, which motivates our endeavor to incorporate WordNet thesauri and domain ontology extraction [23].

5.3 Aligning NFRs

Non-functional requirements (NFRs) are abstract concepts that represent a SPL's architecture drivers [25]. Mismatches in stakeholders' abstraction vocabulary often occur, as an area of surprising controversy. Experts would occasionally misunderstand one another, because they were using the same words in different ways. In fact, experts would sometimes be in "violent agreement" with one another, all the while expressing the same idea in different terms [12]. In our previous work, we tackled terminological interferences by applying the Repertory Grid Technique [30]. One open question is how to come up with a set of primitive elements within the domain of discourse. We would venture an answer of using the FRPs extracted in this work to form a common ground.

Figure 6 shows a sample repertory grid for aligning NFRs in the auto-marker SPL. Five extracted FRPs pro-

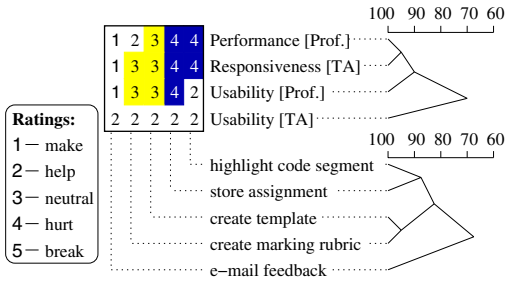

Figure 6. Aligning auto-marker NFRs

vide a shared context for comparing and contrasting abstract NFRs. The clustering analysis of this grid would flag the clash of "Usability" between the professor and the TA's vocabularies due to the low similarity level at 70%. The work on definition hierarchies pointed out that it is acceptable for family members in a SPL to have different interpretations of the same NFR term [25]. We anticipate the mix-and-match of FRPs can help crystallize the varying NFR satisfying criteria; however, we have not yet investigated this idea.

5.4 Driving Design and Development

Teasing out functional requirements facilitates the optimization of quality requirements during the architectural design stages [7]. Behavior-driven development (BDD) [2] has been an evolutionary step toward specifying business-oriented behavior. BDD's motto, "getting the words right", suggests that how we talk about what we are doing influences how we work. The emphasis is on making the system's functionalities explicit with an unambiguous terminology. FRPs capture significant action themes in the domain, and can greatly help BDD to produce a vocabulary that is accurate, accessible, descriptive, and consistent [2].

The OVM rigorously copes with a SPL's variation properties and dependencies, and is orthogonal to traditional requirements, design, code, and test artifacts [32]. Modeling FRPs in OVM allows us to trace the action-oriented concerns throughout the SPL life cycle. In the auto-marker study [1], we were able to relate the OVM to use cases, goal models, and class diagrams, which consistently managed the system functionalities and their relationships.

6 Related Work

Many RE methods have implicitly used the verb-DO pair to model the functional unit: a use case, a primitive requirement [29], a goal concern (task) [26], to name a few. We not only formulate this linguistic clue [34] explicitly in RE, but also introduce an IR-based [27] technique to effectively extract the FRPs from requirements documents.

Domain analysis has been the predominant way of defining a SPL's requirements assets [36]. One of the drawbacks refers to its intrinsic domain dependence. Domain analysis methods count on experts' experience and intuition to manually acquire domain knowledge. Namely, there are no rules that enable engineers to identify domain elements easily [29]. Our approach complements domain analysis by efficiently detecting valuable domain constructs.

The field of feature engineering has worked on modeling domain constructs and their relationships for a number of years [22, 21]. The original proposal envisaged that standard terms (features) used by the stakeholders would naturally emerge at the right level of abstraction, but many studies presented contradictory evidence. For instance, engineers working on the LG's elevator control SPL did not agree on what specific features meant, even after 3 months of domain analysis [12]. The term "feature" therefore remains overused and under-defined in the SPL literature [31]. We focus on the functional characteristic of a feature, and provide a semi-automated way to help search for reusable primitives [29] that can be used for a more complex feature refinement [21].

Liaskos et al. identified variability in goal models via Fillmore's case theory [26]. They took a closer look at the semantic characterization of every goal's OR-decompositions. As was mentioned, a refined goal in [26], expressed mostly by a verb-DO pair, resembled closely the FRPs in our work. Nonetheless, their work illuminated the importance for distinguishing between intentional variability and background variability, and their experience showed that background variability could be effective identified by focusing on the *agentive*, *objective*, and *locational* cases.

Our work also relates to assets modeling techniques like definition hierarchies [25], n-dimensional SPL [35], and PRS [14]. A definition hierarchy is essentially a logical AND tree decomposed by NFRs. The n-dimensional method uses set-theoretic constructs (e.g., subset) to conceptualize variability. A PRS provides a single document to record the common and variable requirements and the decisions pertaining to product derivation. All these approaches impose some hierarchical structure over the family requirements. By their very nature, FRPs are functional so they are better at modeling the SPL's actions and external behaviors. We exploit the OVM [32] to devise the assets that can effect the entire SPL development process.

7 Conclusions

SPLs are rarely created right away but they emerge when a domain becomes mature enough to sustain their long-term investments. A practical adoption pattern is to build a single system, and then build the collection of small variations for the SPL [24]. We contribute an approach to extracting and modeling a SPL's requirements assets by scrutinizing the linguistic characterization of a domain's action-oriented concerns and their variabilities. Studying an auto-marker SPL shows our approach complements domain analysis.

Our work has a number of limitations that we plan to investigate further. First, DO tagging and synonym substitu-

tion need to be incorporated to our current FRP implementation, and the operational cost shall be minimized. Second, sensitivity analysis of domain concepts, as well as domain-specific heuristics production, is in order. Third, integrating more semantic or even structural knowledge will certainly enhance the extraction effectiveness, but the benefits must be weighted against scalability and cost.

Legacy systems and their documentation are valuable source for developing a SPL; yet, their potential remains largely unexploited [20]. The main thrust of our work is to promote a set of low-threshold techniques as a critical enabler for the practitioners to capitalize on the order-of-magnitude improvements offered by SPL engineering.

Acknowledgments. *We are grateful to the CSC340 (Winter 2007) students and staff at U*of*T, and especially thank Jennifer Campbell, Jia Wang, and Sotirios Liaskos for participating in the FAST meetings and helping evaluate the results. Financial support was provided by NSERC.*

References

[1] Archived course website: requirements engineering (winter 2007). http://www.cs.toronto.edu/~nn/csc340h/winter07.

[2] BDD. http://www.behaviour-driven.org.

[3] English stop words. http://en.wikipedia.org/wiki/Stop_word.

[4] Mirrored FTSS. http://web.mit.edu/16.35/www/project.

[5] OpenNLP. http://opennlp.sourceforge.net.

[6] J. Aranda, S. Easterbrook, and G. Wilson. Requirements in the wild: how small companies do it. In *Int'l Reqs Eng Conf*, pages 39–48, New Delhi, India, October 2007.

[7] J. Bosch. *Design & Use Software Architectures: Adopting & Evolving a Product-Line Approach*. Addison-Wesley, 2000.

[8] G. Brunet, M. Chechik, S. Easterbrook, S. Nejati, N. Niu, and M. Sabetzadeh. A manifesto for model merging. In *Int'l Wkshp on Global Integrated Model Mgmt*, 2006.

[9] S. Bühne et al. Exploring the context of product line adoption. In *Int'l Wkshp on Product Family Eng*, pages 19–31, Siena, Italy, November 2003.

[10] J. Carroll et al. High precision extraction of grammatical relations. In *Int'l Wkshp on Parsing Technologies*, 2001.

[11] J. Cleland-Huang et al. The detection and classification of non-functional requirements with applications to early aspects. In *Int'l Reqs Eng Conf*, pages 36–45, Minneapolis, USA, September 2006.

[12] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2001.

[13] J. Dag et al. A linguistic-engineering approach to large-scale requirements management. *IEEE Software*, 22(1):32–39, January/February 2005.

[14] S. R. Faulk. Product-line requirements specification (PRS): an approach and case study. In *Int'l Symp on Reqs Eng*, pages 48–55, Toronto, Canada, August 2001.

[15] C. Fillmore. The case for case. In E. Bach and R. Harms, editors, *Universals in Linguistic Theory*, pages 1–88. New York: Holt, Rinehart and Winston, 1968.

[16] M. Friedewald et al. Status of the software industry in Germany. *Informatik Spektrum*, 24(2):81–90, 2001.

[17] L. Goldin and D. M. Berry. AbstFinder, a prototype natural language text abstraction finder for use in requirements elicitation. *Automated Softw Eng*, 4(4):375–412, October 1997.

[18] J. H. Hayes et al. Advancing candidate link generation for requirements tracing: the study of methods. *IEEE Trans. on Softw Eng*, 32(1):4–19, January 2006.

[19] IEEE Standards Board. IEEE recommended practice for software requirements specifications. 1998.

[20] I. John. Integrating legacy documentation assets into a product line. In *Int'l Wkshp on Product Family Eng*, pages 113–124, Bilbao, Spain, October 2001.

[21] K. Kang et al. FORM: a feature-oriented reuse method with domain-specific reference architectures. *Annals of Softw Eng*, 5:143–168, January 1998.

[22] K. C. Kang et al. Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21, SEI, Carnegie Mellon Univ, November 1990.

[23] L. Kof. Scenarios: identifying missing objects and actions by means of computational linguistics. In *Int'l Reqs Eng Conf*, pages 121–130, New Delhi, India, October 2007.

[24] C. W. Krueger. Easing the transition to software mass customization. In *Int'l Wkshp on Product Family Eng*, pages 282–293, Bilbao, Spain, October 2001.

[25] J. Kuusela and J. Savolainen. Requirements engineering for product families. In *Int'l Conf on Softw Eng*, pages 61–69, Limerick, Ireland, June 2000.

[26] S. Liaskos et al. On goal-based variability acquisition and analysis. In *Int'l Reqs Eng Conf*, pages 76–85, Minneapolis, USA, September 2006.

[27] Y. S. Maarek et al. An information retrieval approach for automatically constructing software libraries. *IEEE Trans. on Softw Eng*, 17(8):800–813, August 1991.

[28] W. J. R. Martin et al. On the processing of a text corpus: from textual data to lexicographic information. In R. R. K. Hartmann, editor, *Lexicography: Principles and Practice*. Academic Press, 1983.

[29] M. Moon, K. Yeom, and H. S. Chae. An approach to developing domain requirements as a core asset based on commonality and variability analysis in a product line. *IEEE Trans. on Softw Eng*, 31(7):551–569, July 2005.

[30] N. Niu and S. Easterbrook. So, you think you know others' goals? A repertory grid study. *IEEE Software*, 24(2):53–61, March/April 2007.

[31] D. L. Parnas. Software product-lines: what to do when enumeration won't work. In *Int'l Wkshp on Variability Modelling of Software-intensive Systems*, 2007.

[32] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, 2005.

[33] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[34] D. Shepherd. *Natural Language Program Analysis*. PhD thesis, Univ of Delaware, 2007.

[35] J. M. Thompson and M. P. Heimdahl. Extending the product family approach to support n-dimensional and hierarchical product lines. In *Int'l Symp on Reqs Eng*, pages 56–64, Toronto, Canada, August 2001.

[36] D. M. Weiss and C. T. R. Lai. *Product-Line Engineering: A Family-Based Software Development Process*. Addison-Wesley, 1999.