Textual Variability Modeling Languages

An Overview and Considerations

Maurice H. ter Beek Klaus Schmid Holger Eichelberger

ISTI–CNR, Pisa, Italy University of Hildesheim, Germany

MODEVAR 2019
Paris, France
09/10/19

Outline

1. Contribution: updated comparison

2. 13 textual variability languages
   - 11 in Schmid & Eichelberger, 2013
   - 3 recent approaches in 2018–2019
   - 4 + 1 dimensions
   - 8 further concerns

3. Conclusion
   - do we need another textual language for feature modeling?

Genesis

Subject: [FML] Proposal for WS contributions
From: David Benavides <benavides@us.es>
Date: 01/03/19, 12:26
To: featuremodellanguage@listas.us.es

Dear all,

I drafted a first version of what could be contributed to the innitiative in general and to the workshop we plan at SPLC 2019 y particular. Of course, this is just a first proposal and can be changed, modified or even completely remade. I propose you to do as follows:

- If you agree on the topic/task given to you, just answer to me with a simple "OK" or something similar.
- If you do not agree on the topic/taks given to you, give an alternative topic/task

I grouped people/topics just by intuition so any change can be done of course.

In addition to that, we will need for the workshop:

- Prepare and circulate a CFPs
- Prepare a workshop website
- Prepare the program
- Interact with SPLC WS chairs.

Please, comment on this

=== TOPIC/TASKS TO BE PREPARED FOR THE SPLC 2019 WORKSHOP ===

- Don Batory/Paul Gazillo: present advances on the usage of KConfig as an example of real world variability modelling tool
- David Benavides/José A. Galindo: present a proposal for a language repository
- Christoph Seidl/ Thomas Thüm/Ina Schaefer: present a proposal for defining language levels in general and for our language in particular
- Camille Salinesi/ Tewfik Ziadi: they are SPLC 2019 general chairs. I don't know if we can ask much more than that.
- Ebrahim Bagheri/ Jules White: A summary of existing proposals
- Kyo Kang, Krzysztof Czarnecki, Klaus Pohl: present a sort of "voices of the experience"
- Klaus Schmid/ Maurice ter Beek : present an updated version of the comparison of existing textual variability modelling languages
- Lidia Fuentes/ Oscar Díaz : present the requirements of a variability modelling language from an architectural perspective
- Mathieu Acher/ Andrzej Wąsowski : developing a reference implementation of the language
- Oystein Haugen: present the experience on CVL and the main points of dicussion during its development
- Patrick Heymans: present the experience developing and maintaining TVL as a textual variability language
- Rick Rabiser: summarize experiences with decision based languages and feature
- Roberto Lopez: present the usage of variability model languages for product synthesis
- Thorsten Berger/ Philippe Collet : Present an analysis of the data collect

Best regards.

David Benavides

Original comparison

Int J Softw Tools Technol Transfer (2015) 17:559–584
DOI 10.1007/s10009-014-0362-x

SPLC 2013

fmt.isti.cnr.it

Mapping the design-space of textual variability modeling languages: a refined analysis

Holger Eichelberger · Klaus Schmid

Published online: 11 December 2014

Abstract Variability modeling is a major part of modern product line engineering. Graphical or table-based approaches to variability modeling are focused around abstract models and specialized tools to interact with these models. However, more recently textual variability modeling languages, comparable to some extent to programming languages, were introduced. We consider the recent trend in product line engineering towards textual variability modeling languages as a phenomenon, which deserves deeper analysis. In this article, we report on the results and approach of a literature survey combined with an expert study. In the literature survey, we identified 11 languages, which enable the textual specification of product line variability and which are sufficiently described for an in-depth analysis. We provide a classification scheme, useful to describe the range of capabilities of such languages. Initially, we identified the relevant capabilities of these languages from a literature survey. The result of this has been refined, validated and partially improved by the expert survey. A second recent phenomenon in product line variability modeling is the increasing scale of variability models. Some authors of textual variability modeling languages argue that these languages are more appropriate for large-scale models. As a consequence, we would expect specific capabilities addressing scalability in the languages. Thus, we compare the capabilities of textual variability modeling techniques, if compared to graphical variability modeling approaches and in particular to analyze their specialized capabilities for large-scale models.

Keywords Variability modeling · Domain-specific languages · Survey · Scalability

1 Introduction

Software product line engineering (SPLE) has been successfully applied in a large range of practical fields and demonstrated to provide very significant economic benefits for companies using it [2]. In general, variability modeling is considered a central part of software product line engineering [3]. A (product line) *variability model* represents an explicit decision of product management about how individual members of a product line differ [4]. A (product line) *variability model* characterizes the properties of variabilities and, in particular, the interrelations among variabilities. Typically, variability management aims at a single source for variability specification and, thus, focuses on the separation of the variability specification from realization models and the implementation of a product line. To serve this purpose, a large range of approaches to variability modeling has been developed. The most frequently used family of approaches to product line variability in industrial modeling [

This article is based on our initial analysis of textual product line variability modeling languages in [1].

H. Eichelberger (✉) · K. Schmid
University of Hildesheim, Hildesheim, Germany
e-mail: eichelberger@sse.uni-hildesheim.de
K. Schmid
e-mail: schmid@sse.uni-hildesheim.de

Original comparison

- *FDL*: mainly a textual representation of feature diagrams
- *Forfamel*: feature modeling in Kumbang (Koalish adds structure)
- *Tree grammars*: Batory's cardinality-based feature models
- *VSL*: feature modeling with configuration links and variable entities
- *SXFM*: XML-based representation of feature models
- *FAMILIAR*: variability modeling (combining and analyzing models)
- *TVL*: textual feature modeling (attributes, cardinalities, modularity)
- μ*TVL*: dropping concepts (adding multiple trees in single model)
- *Clafer*: meta-modeling of classes with feature modeling support
- *VELVET*: extension of TVL (reimplemented from scratch)
- *IVML*: decision modeling (learnability, expressiveness, scalability)

Recent approach

Clafer: Lightweight Modeling of Structure, Behaviour, and Variability

Paulius Juodisius[a], Atrisha Sarkar[b], Raghava Rao Mukkamala[c], Michał Antkiewicz[b], Krzysztof Czarnecki[b], and Andrzej Wąsowski[a]

a IT University of Copenhagen, Denmark

b GSD Lab, University of Waterloo, Canada

c Department of Technology, Kristiania University College, Norway

Abstract Embedded software is growing fast in size and complexity, leading to intimate mixture of complex architectures and complex control. Consequently, software specification requires modeling both structures and behaviour of systems. Unfortunately, existing languages do not integrate these aspects well, usually prioritizing one of them. It is common to develop a separate language for each of these facets.

In this paper, we contribute Clafer: a small language that attempts to tackle this challenge. It combines rich structural modeling with state of the art behavioural formalisms. We are not aware of any other modeling language that seamlessly combines these facets common to system and software modeling.

We show how Clafer, in a single unified syntax and semantics, allows capturing feature models (variability), component models, discrete control models (automata) and variability encompassing all these aspects. The language is built on top of first order logic with quantifiers over basic entities (for modeling structures) combined with linear temporal logic (for modeling behaviour). On top of this semantic foundation we build a simple but expressive syntax, enriched with carefully selected syntactic expansions that cover hierarchical modeling, associations, automata, scenarios, and Dwyer's property patterns.

We evaluate Clafer using a power window case study, and comparing it against other notations that substantially overlap with its scope (SysML, AADL, Temporal OCL and Live Sequence Charts), discussing benefits and perils of using a single notation for the purpose.

ACM CCS 2012

- **Software and its engineering → Formal language definitions; System description languages; Design languages; Specification languages;**

Keywords Language Design, Feature Modeling, Variability Modeling, Behaviour modeling, Semantics

Recent approach

Clafer: Lightweight Modeling of Structure, Behaviour, and Variability

Paulius Juodisius[a], Atrisha Sarkar[b], Raghava Rao Mukkamala[c], Michał Antkiewicz[b], Krzysztof Czarnecki[b], and Andrzej Wąsowski[a]

a IT University of Copenhagen, Denmark

b GSD Lab, University of Waterloo, Canada

c Department of Technology, Kristiania University College, Norway

Abstract Embedded software is growing fast in size and complexity, leading to intimate mixture of complex architectures and complex control. Consequently, software specification requires modeling both structures and behaviour of systems. Unfortunately, existing languages do not integrate these aspects well, usually prioritizing one of them. It is common to develop a separate language for each of these facets.

In this paper, we contribute Clafer: a small language that attempts to tackle this challenge. It combines rich structural modeling with state of the art behavioural formalisms. We are not aware of any other modeling language that seamlessly combines these facets common to system and software modeling.

We show how Clafer, in a single unified syntax and semantics, allows capturing feature models (variability), component models, discrete control models (automata) and variability encompassing all these aspects. The language is built on top of first order logic with quantifiers over basic entities (for modeling structures) combined with linear temporal logic (for modeling behaviour). On top of this semantic foundation we build a simple but expressive syntax, enriched with carefully selected syntactic expansions that cover hierarchical modeling, associations, automata, scenarios, and Dwyer's property patterns.

We evaluate Clafer using a power window case study, and comparing it against other notations that substantially overlap with its scope (SysML, AADL, Temporal OCL and Live Sequence Charts), discussing benefits and perils of using a single notation for the purpose.

ACM CCS 2012

- **Software and its engineering → Formal language definitions; System description languages; Design languages; Specification languages;**

Keywords Language Design, Feature Modeling, Variability Modeling, Behaviour modeling, Semantics

- *Clafer (extended with behavior)*: extends Clafer with a temporal dimension, resulting in a language combining behavior, structure and variability

International Journal of Software Engineering & Applications (IJSEA), Vol.9, No.1, January 2018

PYFML- A TEXTUAL LANGUAGE FOR FEATURE MODELING

A.F. Al Azzawi
Department of Software Engineering, IT Faculty, Philadelphia University, Amman.

ABSTRACT

The Feature model is a typical approach to capture variability in a software product line design and implementation. For that, most works automate feature model using a limited graphical notation represented by propositional logic and implemented by Prolog or Java programming languages. These works do not properly combine the extensions of classical feature models and do not provide scalability to implement large size problem issues. In this work, we propose a textual feature modeling language based on Python programming language (PyFML), that generalizes the classical feature models with instance feature cardinalities and attributes which be extended with highlight of replication and complex logical and mathematical cross-tree constraints. textX Meta-language is used for building PyFML to describe and organize feature model dependencies, and PyConstraint Problem Solver is used to implement feature model variability and its constraints validation. The work provides a textual human-readable language to represent feature model and maps the feature model descriptions directly into the object-oriented representation to be used by Constraint Problem Solver for computation. Furthermore, the proposed PyFML makes the notation of feature modeling more expressive to deal with complex software product line representations and using PyConstraint Problem Solver.

KEYWORDS

Software Product Line, Feature Model, Python Programming language, Domain-Specific Language, Constraint Programming

1. INTRODUCTION

A software product line (SPL) is considered as a set of products that all shared a lot of characteristics and vary in their particular configuration of variability, portfolios of very similar products are developed to get lower product cost, faster development cycles, and better quality [1], while Software product line engineering (SPLE) means to develop up a group of similar software products where commonality and variability among the family members (product variations) are explicitly determined by features where each feature is a visible characteristic of product in SPL related to product configuration[2]. Modeling variability is an essential in SPLE and a typical approach of the variability is called feature modeling and presents feature models as feature diagrams and the graphical notations is the most popular type of representation for feature diagrams [2], [3], particularly it encourages analysis to present a graphical language for these feature diagrams and associate it with formal representation considered as a base for building software engineering tool.

As far back as the first presentation of feature models [4], an extraordinary number of expansions

International Journal of Software Engineering & Applications (IJSEA), Vol.9, No.1, January 2018

PYFML- A TEXTUAL LANGUAGE FOR FEATURE MODELING

A.F. Al Azzawi
Department of Software Engineering, IT Faculty, Philadelphia University, Amman.

ABSTRACT

The Feature model is a typical approach to capture variability in a software product line design and implementation. For that, most works automate feature model using a limited graphical notation represented by propositional logic and implemented by Prolog or Java programming languages. These works do not properly combine the extensions of classical feature models and do not provide scalability to implement large size problem issues. In this work, we propose a textual feature modeling language based on Python programming language (PyFML), that generalizes the classical feature models with instance feature cardinalities and attributes which be extended with highlight of replication and complex logical and mathematical cross-tree constraints. textX Meta-language is used for building PyFML to describe and organize feature model dependencies, and PyConstraint Problem Solver is used to implement feature model variability and its constraints validation. The work provides a textual human-readable language to represent feature model and maps the feature model descriptions directly into the object-oriented representation to be used by Constraint Problem Solver for computation. Furthermore, the proposed PyFML makes the notation of feature modeling more expressive to deal with complex software product line representations and using PyConstraint Problem Solver.

KEYWORDS

Software Product Line, Feature Model, Python Programming Language, Domain-Specific Language, Constraint Programming

1. INTRODUCTION

A software product line (SPL) is considered as a set of products that all shared a lot of characteristics and vary in their particular configuration of variability, portfolios of very similar products are developed to get lower product cost, faster development cycles, and better quality [1], while Software product line engineering (SPLE) means to develop up a group of similar software products where commonality and variability among the family members (product variations) are explicitly determined by features where each feature is a visible characteristic of product in SPL related to product configuration[2]. Modeling variability is an essential in SPLE and a typical approach of the variability is called feature modeling and presents feature models as feature diagrams and the graphical notations is the most popular type of representation for feature diagrams [2], [3], particularly it encourages analysis to present a graphical language for these feature diagrams and associate it with formal representation considered as a base for building software engineering tool.

As far back as the first presentation of feature models [4], an extraordinary number of expansions

- *PyFML*: a textual feature modeling language based on Python

Software Qual J (2019) 27:307–347
https://doi.org/10.1007/s11219-017-9400-8

Modeling variability in the video domain: language and experience report

Mauricio Alférez[1] · Mathieu Acher[2] · José A. Galindo[3] · Benoit Baudry[4] · David Benavides[3]

Published online: 12 March 2018

Abstract In an industrial project, we addressed the challenge of developing a software-based video generator such that consumers and providers of video processing algorithms can benchmark them on a wide range of video variants. This article aims to report on our positive experience in modeling, controlling, and implementing software variability in the video domain. We describe how we have designed and developed a variability modeling language, called VM, resulting from the close collaboration with industrial partners during 2 years. We expose the specific requirements and advanced variability constructs; we developed and used to characterize and derive variations of video sequences. The results of our experiments and industrial experience show that our solution is effective to model complex variability information and supports the synthesis of hundreds of realistic video variants. From the software language perspective, we learned that basic variability mechanisms are useful but not enough; attributes and multi-features are of prior importance; meta-information and

✉ Mathieu Acher
mathieu.acher@irisa.fr

Recent approach

Software Qual J (2019) 27:307–347
https://doi.org/10.1007/s11219-017-9400-8

Modeling variability in the video domain: language and experience report

Mauricio Alférez[1] · Mathieu Acher[2] · José A. Galindo[3] · Benoit Baudry[4] · David Benavides[3]

- *Variability Modeling (VM)*: a language developed in an industrial project, with specific constraints to ease reasoning particularly for applications in the video domain

Published online: 12 March 2018

Abstract In an industrial project, we addressed the challenge of developing a software-based video generator such that consumers and providers of video processing algorithms can benchmark them on a wide range of video variants. This article aims to report on our positive experience in modeling, controlling, and implementing software variability in the video domain. We describe how we have designed and developed a variability modeling language, called VM, resulting from the close collaboration with industrial partners during 2 years. We expose the specific requirements and advanced variability constructs; we developed and used to characterize and derive variations of video sequences. The results of our experiments and industrial experience show that our solution is effective to model complex variability information and supports the synthesis of hundreds of realistic video variants. From the software language perspective, we learned that basic variability mechanisms are useful but not enough; attributes and multi-features are of prior importance; meta-information and

✉ Mathieu Acher
mathieu.acher@irisa.fr

Dimensions

- Configurable elements
  - forms of variation (optional, alternative, multiple, extension)
  - attached information
  - cardinalities
  - references
  - additional data types (predefined, user-defined, derived)
- Constraint support
  - expressions (simple, propositional, first-order, relational, arithmetic)
- Configuration support
  - default values, value assignment
  - partial or complete configurations
- Scalability support
  - composition (i.e., capability of integrating units of configurable elements into a single model)
- Formal semantics

Updated comparison

fmt.isti.cnr.it

	forms of variation							data types			constraint expressions					configurations					
Language	optional	alternative	multiple	extension	attached info	cardinalities	references	predefined	derived	user-defined	simple	propositional	first-order	relational	arithmetic	default values	assign values	partial	complete	composition	formal semantics
FDL	+	+	+	−	−	±	−	−	−	−	+	−	−	−	−	+	−	−	−	−	+
Forfamel	+	+	+	+	+	+	+	−	−	+	−	+	+	+	+	−	+	−	+	−	±
Tree grammars	+	+	+	−	−	+	−	−	−	−	−	+	−	−	−	−	−	−	−	−	−
VSL	+	+	+	+	+	+	+	+	−	+	+	?	?	−	?	+	+	+	+	+	−
SXFM	+	+	+	−	−	+	−	−	−	−	−	?	−	−	−	−	−	−	−	−	−
FAMILIAR	+	+	+	−	−	−	?	+	+	−	−	+	−	−	−	−	+	+	+	+	−
TVL	+	+	+	?	+	+	+	+	−	+	−	+	−	+	+	−	+	−	−	−	+
μTVL	+	+	±	+	+	+	−	+	−	−	+	+	−	+	+	−	?	+	+	−	+
Clafer	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	−	+	+	+	+	+
VELVET	±	+	+	+	+	±	+	+	−	−	−	+	−	+	−	+	+	+	±	+	−
IVML	+	+	+	+	+	±	+	+	+	+	−	+	+	+	+	+	+	+	±	+	−
PyFML	+	+	+	−	+	−	−	+	−	−	+	+	−	+	+	+	+	−	−	−	−
VM	+	+	+	−	+	+	−	+	−	−	+	+	−	+	+	+	+	+	±	+	−

direct (+), indirect (±), unclear (?) or no (−) support

Further concerns

- First citizen concept
  - stuck, as mostly centered on features
  - other approaches (e.g., Clafer, IVML) use different main elements
  - benefits: richness of expression, typing, etc.
- Quantitative variability modeling
  - quantification is form of variability (first citizen concept!)
  - relegated to feature attributes not natural
- Ecosystem support
  - if ecosystems are a natural extension of PLE, then why are they mostly ignored?
  - openness of variation lacks as a concept
- 'Exotic' modeling capabilities to address known shortcomings
  - modeling topologies
  - integration of behavior

Further concerns

- Binding time
  - fundamental, but ignored!?
  - other forms of meta-variability
- Analyzability
  - different classes of variability match different classes of analysis
  - customize classes (explicit and selectable)
- Extensible language design
  - capabilities to extend a language or embed concepts into host language (DSLs)
  - support experimentation
- Modular language design (language levels)
  - *one size fits all* impossible
  - different needs only through modular design
  - support experimentation and reuse

Conclusion

- Updated comparison of the main characteristics of 13 textual variability modeling languages
- Overview of the currently available textual variability modeling languages (did we forget some?)

So, do we need another textual language for feature modeling?

- No need to start from scratch, we rather need more innovative (textual) variability modeling approaches
- In particular, an extensible and modular language design with sub-languages catering for different needs, holistically integrated into an over-arching concept

Conclusion

- Updated comparison of the main characteristics of 13 textual variability modeling languages
- Overview of the currently available textual variability modeling languages (did we forget some?)

So, do we need another textual language for feature modeling?

- No need to start from scratch, we rather need more innovative (textual) variability modeling approaches
- In particular, an extensible and modular language design with sub-languages catering for different needs, holistically integrated into an over-arching concept

Conclusion

- Updated comparison of the main characteristics of 13 textual variability modeling languages
- Overview of the currently available textual variability modeling languages (did we forget some?)

So, do we need another textual language for feature modeling?

- No need to start from scratch, we rather need more innovative (textual) variability modeling approaches
- In particular, an extensible and modular language design with sub-languages catering for different needs, holistically integrated into an over-arching concept