



Elder José Reoli Cirilo

**Supporting Heterogeneous Configuration
Knowledge of Software Product Lines**

Tese de Doutorado

Thesis presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática do Centro Técnico Científico da PUC-Rio, as partial fulfillment of the requirements for the degree of Doutor.

Advisor: Prof. Carlos José Pereira de Lucena

Rio de Janeiro
April 2012



Elder José Reoli Cirilo

Supporting Heterogeneous Configuration Knowledge of Software Product Lines

Thesis presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática do Centro Técnico Científico da PUC-Rio, as partial fulfillment of the requirements for the degree of Doutor.

Prof. Carlos José Pereira de Lucena

Advisor

Departamento de Informática — PUC-Rio

Prof. Alessandro Fabricio Garcia

Departamento de Informática – PUC-Rio

Prof. Arndt Von Staa

Departamento de Informática – PUC-Rio

Prof. Paulo Henrique Monteiro Borba

Centro de Informática – UFPE

Prof. Vander Ramos Alves

Departamento de Ciência da Computação – UNB

Prof. José Eugenio Leal

Coordinator of the Centro Técnico Científico — PUC-Rio

Rio de Janeiro — April 10, 2012

All rights reserved.

Elder José Reoli Cirilo

Doctoral and Master's Degree in Computer Science from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio) and Bachelor's Degree in Computer Science from the Federal University of Juiz de Fora (UFJF).

Bibliographic data

Cirilo, Elder

Supporting Heterogeneous Configuration Knowledge of Software Product Lines / Elder José Reoli Cirilo; advisor: Carlos José Pereira de Lucena - 2012.

2 v. 104 f: il. ; 30 cm

Tese (Doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2012

Inclui bibliografia.

1. Informática – Teses. 2. Linhas de Produtos de Software. 3. Frameworks Orientados a Objetos. 4. Linguagens Específicas de Domínio. 5. Ferramentas para Linhas de Produtos de Software. I. Lucena, Carlos. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

First, I give thanks to God for everything and for my beloved family. To my dear parents Maria Josefina and José Cyrilo for supporting me since I was born. To my dearest aunt, "Tia Santa", for always loving and supporting me. Even living so far during my PhD, they were always very present. To Suelen for his love and for supporting me along the PhD journey. God helped us to overcome the difficulties inherent to PhDs – she was also doing her PhD at the same time. To my sisters Elisana and Elaina.

To my advisor, Carlos Lucena, who was always very supportive, and whose ideas were great inspiration to this PhD work. To Professors Uirá Kulesza and Alessandro Garcia for the great discussions on software product line.

To my friend and PhD colleague Ingrid Nunes who contributed on the empirical study performed in this work. To all my friends and PhD colleagues who also contributed with great discussions. To the LES research group in general for the friendship.

To the professors of the committee, Paulo Borba, Vander Alves, Arndt von Staa and Alessandro Garcia for giving great suggestions on how to improve this thesis.

To the Informatics Department, its professors and the technical staff, for supporting my research, and to CNPq, CAPES and FAPERJ for funding it.

Resumo

Cirilo, Elder; Lucena, Carlos; Advisor. **Suportando Conhecimento de Configuração Heterogêneo de Linhas de Produtos de Software.** Rio de Janeiro, 2012. 104p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Softwares personalizados para as necessidades de um cliente proveem vantagem competitiva quando comparados aos softwares de propósito geral. Linhas de produtos de softwares tem como objetivo a produção de produtos personalizados a partir de um conjunto de *features* reutilizáveis. É conhecido que o reuso sistemático de *features* potencialmente leva a ganhos significativos como rápida produção de software a um baixo custo e com maior qualidade. Na prática, a construção de linhas de produtos de softwares empresariais pode se tornar uma tarefa arriscada e sujeita a erros quando se leva em consideração o uso das técnicas atuais de implementação de *features*. O desenvolvimento de linhas de produtos de software empresariais de fato requer a convergência coordenada de várias visões (ex., especialistas de domínio, projetistas de interface, e desenvolvedores). Neste caso, cada participante do processo de desenvolvimento tem uma linguagem particular para resolver o problema específico a sua especialidade. Os desafios para integração de diferentes linguagens, evitando uma potencial cacofonia, é o problema do conhecimento de configuração heterogêneo. Nesta tese, nos examinamos as dificuldades atuais na especificação do conhecimento de configuração heterogêneo e como solução nos propomos a noção de Linguagens de Modelagem do Conhecimento do Domínio (LMCD). O propósito das LMCDs é evidenciar os conceitos do domínio e suas interfaces de programação, o que ajuda reduzir o ofuscamento do código fonte e aumentar a compreensão. Além disso, evidenciando os conceitos específicos de domínio, somos aptos a prevenir inconsistências em produtos pela detecção de erros em toda a linha de produto de software. Outro resultado deste trabalho de pesquisa é GenArch⁺, uma ferramenta extensível que suporta de forma flexível a incorporação de LMCDs na engenharia de linhas de produtos de softwares tradicional. Nos enfatizamos os benefícios da ferramenta, incluindo simplicidade, expressividade, e a capacidade de ser independente de qualquer tecnologia de implementação do domínio. Finalmente, nos ilustramos e avaliamos o uso de LMCDs em três diferentes linhas de produtos de software.

Palavras-chave

Linhas de Produtos de Software; Frameworks Orientados a Objetos; Linguagens Específicas de Domínio; Ferramentas para Linhas de Produtos de Software;

Abstract

Cirilo, Elder; Lucena, Carlos; Advisor. **Supporting Heterogeneous Configuration Knowledge of Software Product Lines**. Rio de Janeiro, 2012. 104p. DSc Thesis — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Customer-specific software can provide a competitive advantage compared to general-purpose software. A software product line (SPL) aims at tailoring customer-specific products from a set of reusable features. It is well known that systematic reuse of features potentially leads to significant gains such as faster production with low costs and high quality. In practice, building enterprise software product lines might become a laborious and error-prone task when we take into account the use of current implementation techniques. Successful development of modern enterprise software product lines requires the convergence of multiple views (e.g., domain experts, interface designers, and developers), which are usually managed via domain-specific concepts. In this case, each participant of the development process has a particular working language to solve specific problems to its expertise. The challenge of integrating these differences into the engineering process, avoiding the potential cacophony of multiple different languages, is the heterogeneous configuration knowledge problem. In this thesis, we examine the current shortcomings on the specification of heterogeneous configuration knowledge and as a solution we propose the notion of Domain Knowledge Modeling Languages (DKMLs). The purpose of DKMLs is to put in evidence the domain concepts and their programming interfaces, which help to reduce source code obfuscation and increase feature comprehension. Moreover, evidencing the domain-specific concepts, we are also able to prevent inconsistencies on products by detecting errors in the entire software product line. Another result of this research is the GenArch⁺, an extensible tool that supports the flexible incorporation of DKMLs in traditional software product line engineering. We emphasize the benefits of GenArch⁺, including simplicity, expressiveness, and being framework independent. Finally, we illustrate and evaluate the use of DKMLs in three different product lines.

Keywords

Software Product Lines; Object-oriented Frameworks; Domain-specific Languages; Software Product Lines Tools;

Table of Contents

1	Introduction	10
1.1	Contributions	14
1.2	Outline of the Thesis Structure	15
2	Problem Formulation	17
2.1	Scenarios when Assigning Framework-based Software Product Lines Source-code to Features	22
2.1.1	Configuration and Customization Code	22
2.1.2	Cross-references between Concept Instances	25
2.1.3	References to Concept Instances inside Customization Code	27
2.1.4	Context Sensitive Instantiation Constraints	28
2.2	Limitations of the Related Work	29
2.2.1	Annotation-based Techniques	30
2.2.2	Model-based Techniques	32
2.3	Summary and Goals	35
3	Supporting Heterogeneous Configuration Knowledge of Software Product Lines with Domain Knowledge Modeling Languages	37
3.1	Domain-specific model-supported Engineering of Framework-based Software Product Lines	38
3.2	DKML: Definition and Properties	40
3.3	Visualizing Features Code	46
3.4	Consistency Checking and Guidance	51
3.4.1	Disciplined Feature Assignment	51
3.4.2	Guidance	53
3.4.3	Consistency Checking	55
3.5	Summary	58
4	GenArch ⁺ : Building Software Product Lines with Domain Knowledge Modeling Languages	59
4.1	Tool Architecture Overview	59
4.1.1	Domain Knowledge Schema	60
4.2	DKML Editor and Views	62
4.3	A Language For Reverse Engineering Domain Knowledge Models	65
4.3.1	Reverse Engineering Domain Knowledge Models	67
4.4	Summary	69
5	Evaluation	70
5.1	Selected Product Lines	70
5.2	Assessment of Modularity and Complexity	72
5.2.1	Study Phases and Analysis Procedures	72
5.2.2	Quantitative Metrics Suite	73
5.2.3	Results: Separation of Concerns	74
5.2.4	Results: Size	75

5.3	Empirical Evaluation	76
5.3.1	Experiment Hypotheses	77
5.3.2	Background of the Participants	77
5.3.3	Experimental Design	78
5.3.4	Variables and Analysis	80
5.3.5	First Controlled Experiment: Results, Analysis and Discussion	80
5.3.6	Second Controlled Experiment: Analysis and Discussion	85
5.4	Threats to Validity	93
5.5	Summary	94
6	Final Remarks and Future Work	96
6.1	Limitations and Future Work	98

List of Figures

2.1	Source code instantiating Spring-provided concepts and their assignment to features.	18
2.2	Source code instantiating Jadex-provided concepts and their assignment to features.	19
2.3	Configuration and customization Code.	23
2.4	Spring constructor-based dependency injection.	24
2.5	Cross-references between concept instances.	25
2.6	Source code of mutually exclusive concepts instances.	26
2.7	References to concept instances inside customization code.	27
2.8	Context sensitive instantiation constraints.	29
2.9	WeatherService instantiation code annotated with features.	31
2.10	User agent and Event Scheduler capability instantiation codes annotated with features.	32
2.11	Software product line architecture defined in pure::variants.	33
3.1	Ecore language structure.	41
3.2	Exemplar Spring Domain Knowledge Modeling Language.	42
3.3	Exemplar Spring Domain Knowledge Model.	43
3.4	Spring Domain Knowledge Modeling Language with references.	44
3.5	Spring Domain Knowledge Model with references.	45
3.6	Spring Domain Knowledge Modeling - <i>Bean</i> EventDAO.	48
3.7	Detailed creation of Spring-DKM following the guidance rules.	54
4.1	GenArch ⁺ architectural overview.	59
4.2	Domain Knowledge Schema	61
4.3	GenArch ⁺ reflective DKML editor.	62
4.4	Filtered exemplar Spring-DKML	63
4.5	WeatherService occurrence marked.	63
4.6	Reference View: references to the <i>Bean</i> WeatherService	64
4.7	Usage View: usage of the <i>Belief</i> client	64
4.8	Spring-DKML mappings patterns.	68
5.1	Box plot for correctness.	81
5.2	Box plot for time.	82
5.3	Measure of Correct Answers	84
5.4	Measure of Correct Answers and Expertise	85
5.5	Box plot for correctness.	86
5.6	Box plot for time.	88
5.7	Measure of correct answers per tasks.	90
5.8	Measure of time per tasks.	91
5.9	Box plot for difficult - Tasks T1 and T2.	92
5.10	Box plot for difficult - Tasks T3 and T4.	93

List of Tables

2.1	Results of computed metrics.	21
2.2	Benefits and limitations of annotation and model-based techniques.	36
4.1	Mapping Types	66
5.1	Main characteristics of the target product lines.	71
5.2	Configuration Knowledge Modularization – separation of concerns metrics.	74
5.3	Configuration Knowledge Modularization – size metrics.	75
5.4	Degree of Expertise of the participants	78
5.5	Description of the second comprehension questionnaire.	79
5.6	Latin Square configuration	79
5.7	ANOVA test for participants time.	81
5.8	Descriptive statistics of the experimental results.	87
5.9	ANOVA test for participants answers.	87
5.10	Tukey's HSD multiple comparison test results for answers.	87
5.11	Descriptive statistics (Answers and Time) - Task 1	90
5.12	Descriptive statistics (Answers and Time) - Task 2	91
5.13	Descriptive statistics (Answers and Time) - Task 3	92
5.14	Descriptive statistics (Answers and Time) - Task 4	93