

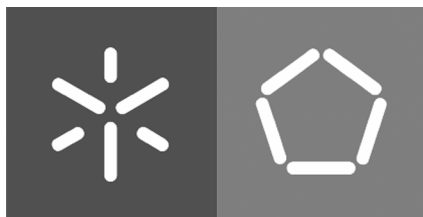


Universidade do Minho
Escola de Engenharia

Sofia Manuela Fevereiro de Azevedo

**UML Metamodelling and ERP Software
Solutions: Experiments with Microsoft DSL
Tools**

Maio de 2008



Universidade do Minho

Escola de Engenharia

Sofia Manuela Fevereiro de Azevedo

**UML Metamodelling and ERP Software
Solutions: Experiments with Microsoft DSL
Tools**

Dissertação de Mestrado em

Sistemas de Informação

Trabalho efectuado sob a orientação do

Professor Doutor Ricardo J. Machado

Departamento de Sistemas de Informação

Maio de 2008

DECLARAÇÃO

Nome: SOFIA MANUELA FEVEREIRO DE AZEVEDO

Endereço Electrónico: sofiaazo@gmail.com

Telefone: +351 912 959 244

Bilhete de Identidade: 12450574

Título da Dissertação de Mestrado: UML Metamodelling and ERP Software Solutions: Experiments with Microsoft DSL Tools

Orientador: Professor Doutor Ricardo J. Machado

Data de Conclusão: Maio de 2008

Designação do Mestrado: Mestrado em Sistemas de Informação

É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA TESE/TRABALHO, APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE.

Universidade do Minho, ____ / ____ / _____

Assinatura: _____

Para os que me trouxeram até aqui...

Abstract

Microsoft DSL (Domain-Specific Language) Tools allow the definition at the metamodelling level of graphical languages suited to a particular domain. The DSL Tools also allow the conception of models with those graphical languages. The proof of concept reported in this dissertation focuses on the domain of a part of the Primavera ERP (Enterprise Resource Planning) software solution. It exposes a metamodelling approach which can be followed when using the tool to model visual domain-specific languages. It includes a stereotyping approach, abstract and concrete syntaxes' setting down. The stereotypes allow the adaptation of the graphical language to the domain. Together, stereotypes and language definition through a metamodel make up the DSVL (Domain-Specific Visual Language). This dissertation explains how to perform both the abstract syntax design through metamodels resembling UML (Unified Modelling Language) class diagrams and the concrete syntax definition through the mapping between the elements in the abstract syntax and the visual constructs of the DSVL. Having metamodels inspired by UML is a pertinent approach defended in this dissertation. UML is a standard with worldwide impact, therefore, graphical languages inspired by UML can be handled by professionals worldwide to design their applications and communicate their design decisions. We can create UML-based graphical languages with Microsoft DSL Tools in order to be able to reason about the solution to the problem domain of a portion of the Primavera ERP and still be able to communicate with professionals familiarized with UML about our design decisions. A compromise between domain knowledge and cross-domain knowledge is established with a UML-inspired language tailored to a specific domain. In this dissertation, stereotypes and domain-specific concepts tailor the graphical languages to the domain, whereas metamodels determine a UML-based syntax for the graphical languages.

Resumo

As Microsoft DSL (*Domain-Specific Language*) Tools permitem a definição ao nível da metamodelação de linguagens gráficas ajustadas a um domínio em particular. As DSL Tools também permitem a concepção de modelos expressos nessas mesmas linguagens. A prova de conceito reportada nesta dissertação foca-se no domínio de uma parte da solução de software Primavera ERP (*Enterprise Resource Planning*). A dissertação expõe uma abordagem de metamodelação que pode ser seguida durante a utilização da ferramenta DSL Tools para modelar linguagens visuais específicas de um domínio. Inclui uma abordagem de estereotipagem, bem como a definição de sintaxes abstracta e concreta. Os estereótipos permitem a adaptação da linguagem gráfica ao domínio. Juntos, os estereótipos e a definição da linguagem através de um metamodelo constituem a DSVL (*Domain-Specific Visual Language*). Esta dissertação explica como executar tanto o design da sintaxe abstracta através de metamodelos que se assemelham a diagramas de classes UML (*Unified Modelling Language*), como a definição da sintaxe concreta através do mapeamento entre os elementos da sintaxe abstracta e os elementos visuais da DSVL. Construir metamodelos inspirados pela UML é uma abordagem pertinente defendida nesta dissertação. A UML é um standard com impacto mundial, logo, linguagens gráficas inspiradas pela UML podem ser manuseadas por profissionais a nível mundial para desenhar as suas aplicações e comunicar as suas decisões de desenho. Podem ser criadas linguagens gráficas baseadas em UML com as Microsoft DSL Tools com o intuito de tornar possível o raciocínio acerca da solução para o domínio do problema de uma porção do ERP Primavera e mesmo assim ser possível comunicar com profissionais familiarizados com a UML acerca das decisões de desenho tomadas. Um compromisso entre o conhecimento do domínio e o conhecimento que é transversal a vários domínios é estabelecido com uma linguagem inspirada em UML e talhada para um domínio específico. Nesta dissertação, os estereótipos e os conceitos específicos do domínio adaptam as linguagens gráficas ao domínio, enquanto que os metamodelos determinam uma sintaxe baseada em UML para as mesmas linguagens gráficas.

Acknowledgements

I would like to express my appreciation to Primavera Business Software Solutions, S.A. that gave context to this dissertation.

I would like to express particular gratitude to my mentor, Professor Doutor Ricardo J. Machado, who believed I was able to perform this work, who gave me all the necessary guidance and who was always available to interact.

At last I would like to express my appreciation to all those who contributed somehow to the execution of this dissertation and whose names I don't mention above.

Contents

Abstract	v
Resumo	vii
Acknowledgements	ix
1. Introduction	1
1.1. Software Development with DSLs	1
1.2. The Advantages of Using DSLs Embedded in a Metamodelling Approach	3
1.3. Research Goals	5
1.4. The Demonstration Case	5
1.5. Dissertation Roadmap	7
2. Concepts to Understand DSLs	9
2.1. Introduction	9
2.2. MDD and Metamodelling	10
2.3. Transformations and Software Evolution	16
2.4. Software Development Methodologies and DSLs	19
2.4.1. 4SRS	20
2.4.2. VA	21
2.4.3. FAST	22
2.4.4. SF and DSLs	23
2.5. Conclusions	28
3. Metamodelling and Modelling Environments as Tools to Handle DSLs	31
3.1. Introduction	31
3.2. Development Cycle of Metamodelling Environments	32
3.3. DSLs from a Metamodelling Perspective	34
3.4. Using Modelling Environments to Configure Instances of UML Concepts with Domain-Specific Concepts	44
3.5. Conclusions	48
4. From Metamodelling DSLs Inspired by UML to Designing Domain-Specific Models	51
4.1. Introduction	51
4.2. First Experimentation with Microsoft DSL Tools	54
4.2.1. <i>The Example of Two Graphical Languages: Class Diagram and State Machine Diagram</i>	54
4.2.2. <i>The Generated Code</i>	59
4.3. Metamodelling with Microsoft DSL Tools Considering the UML Superstructure	62
4.3.1. <i>Mapping Some of DSL Tools' Concepts into UML Metamodel's Concepts</i>	62
4.3.2. <i>Metamodel for the Sales Domain of the Primavera ERP Solution</i>	65
4.4. Discussing the Undertaken Metamodelling Approach and the Conceived DSLs	77
4.5. Conclusions	81
5. Conclusions	83
5.1. Results Analysis	83
5.2. Future Work	86
References	89
Appendix I. Metamodel Configuration Offered by Microsoft DSL Tools	93

List of Figures

Figure 1 – Some business objects of the Primavera ERP	6
Figure 2 – Example of the business object product	7
Figure 3 – OMG modelling infrastructure or Four-Layer Architecture of UML	14
Figure 4 – Development cycle of metamodelling environments	33
Figure 5 – Equivalence between DSLs and stereotypes	35
Figure 6 – Representation of the concepts actor, use case, extend and include used to model use cases (from the UML superstructure).....	36
Figure 7 – Representation of the concept class used to model classes (from the UML superstructure).....	37
Figure 8 – Representation of the concept association class used to model classes (from the UML superstructure).....	38
Figure 9 – Representation of the concepts dependency and realization used to model classes (from the UML superstructure).....	39
Figure 10 – Representation of the concept generalization used to model classes (from the UML superstructure).....	39
Figure 11 – Representation of the concept interface used to model classes (from the UML superstructure).....	40
Figure 12 – Representation of the concept action used to model activities (from the UML superstructure).....	40
Figure 13 – Representation of the concepts activity final node and initial node used to model activities (from the UML superstructure)	41
Figure 14 – Representation of the concept control flow used to model activities (from the UML superstructure).....	41
Figure 15 – Representation of the concept activity partition used to model activities (from the UML superstructure).....	42
Figure 16 – Representation of the concepts fork node, join node, merge node and decision node used to model activities (from the UML superstructure).....	43
Figure 17 – Representation of the concepts state, pseudostate, final state and transition used to model state machines (from the UML superstructure)	44
Figure 18 – Stereotyping metamodelling hierarchy.....	48
Figure 19 – Transform All Templates button	52
Figure 20 – The DSL Designer	53
Figure 21 – The DSL Experimental Designer	53
Figure 22 – Layered modelling architecture of Microsoft DSL Tools	54
Figure 23 – The Primavera ERP general business objects domain model	56
Figure 24 – Domain model for a possible state machine diagram regarding the Primavera ERP general business objects.....	57
Figure 25 – Example of a model equivalent to a class diagram	58
Figure 26 – Example of a model equivalent to a state machine diagram	58
Figure 27 – Layered modelling architecture of Microsoft DSL Tools for the first experimentation with the tool	59
Figure 28 – Excerpt of the file DomainClasses.cs containing generated code for the domain classes in the domain model of the Primavera ERP general business objects.....	60
Figure 29 – Excerpt of the file DomainRelationships.cs containing generated code for the domain relationships in the domain model of the Primavera ERP general business objects	60
Figure 30 – Code that defines the domain property Name of the class BusinessObject	60

Figure 31 – Code that defines the domain property BusinessObjects	61
Figure 32 – Excerpt of the file DomainClasses.cs containing generated code for the domain classes in the domain model of a possible state machine diagram regarding the Primavera ERP general business objects’	61
Figure 33 – Excerpt of the file DomainRelationships.cs containing generated code for the domain relationships in the domain model of a possible state machine diagram regarding the Primavera ERP general business objects’	62
Figure 34 – Metamodelling environment’s Toolbox	63
Figure 35 – Modelling environment’s Toolbox	63
Figure 36 – Use case metamodel that is part of the Primavera ERP metamodel for the sales domain.....	66
Figure 37 – Example of a use case diagram in the context of sales.....	67
Figure 38 – Class metamodel that is part of the Primavera ERP metamodel for the sales domain.....	69
Figure 39 – Example of a class diagram in the context of sales	71
Figure 40 – Activity metamodel that is part of the Primavera ERP metamodel for the sales domain.....	73
Figure 41 – Example of an activity diagram (sell product) in the context of sales	74
Figure 42 – State machine metamodel that is part of the Primavera ERP metamodel for the sales domain.....	76
Figure 43 – Example of a state machine diagram (product) in the context of sales	77
Figure 44 – Division of roles along the layered modelling architecture of Microsoft DSL Tools	78
Figure 45 – PIM and PSM levels across the layered modelling architecture of Microsoft DSL Tools	79
Figure 46 – File containing the definition of the DSL and files containing generated code ...	93
Figure 47 – Experimental Designer’s Toolbox notational elements	94
Figure 48 – Diagram associated with the Experimental Designer	94
Figure 49 – Class represented by the diagram	95
Figure 50 – Property Position	95
Figure 51 – Mapping of a domain class to a shape.....	96
Figure 52 – Mapping of a domain class attribute to a decorator	96
Figure 53 – DSL Designer’s Toolbox.....	97
Figure 54 – Properties of an element from the Experimental Designer’s Toolbox.....	97
Figure 55 – Inheritance Modifier property	98
Figure 56 – Definition of a base shape for a connector	98
Figure 57 – Property Allows Duplicates.....	99

List of Tables

Table 1 – Fundamental forms of change in software artefacts	12
Table 2 – Key concepts of model transformation	17
Table 3 – Strategies to design DSLs	26
Table 4 – Types of DSL transformations considering abstraction as a variable	34
Table 5 – Common syntax of two possible Primavera ERP concepts for the sales domain....	46
Table 6 – Example of specialized syntax of two possible Primavera ERP concepts for the sales domain.....	46
Table 7 – Mapping between some types of elements from the UML metamodel and the types of elements from the metamodel of DSL Tools.....	64

Acronyms

4SRS	<i>Four Step Rule Set</i>
API	<i>Application Programming Interface</i>
CAE	<i>Computer-Aided Engineering</i>
CASE	<i>Computer-Aided Software Engineering</i>
CIM	<i>Computation Independent Model</i>
DSDE	<i>Domain-Specific Design Environment</i>
DSME	<i>Domain-Specific Metamodelling Environment</i>
DSL	<i>Domain-Specific Language</i>
DSVL	<i>Domain-Specific Visual Language</i>
EJB	<i>Enterprise JavaBeans</i>
ERP	<i>Enterprise Resource Planning</i>
FAST	<i>Family-Oriented Abstraction, Specification and Translation</i>
FMOTS	<i>Functional Modules Off-The-Shelf</i>
FODA	<i>Feature-Oriented Domain Analysis</i>
GME	<i>Generic Modelling Environment</i>
GPL	<i>General Programming Language</i>
GUI	<i>Graphical User Interface</i>
HTML	<i>HyperText Markup Language</i>
IIS	<i>Industrial Information System</i>
MDA	<i>Model-Driven Architecture</i>
MDD	<i>Model-Driven Development</i>
MOF	<i>Meta-Object Facility</i>
OCL	<i>Object Constraint Language</i>
OMG	<i>Object Management Group</i>
OO	<i>Object-Oriented</i>
PBSS	<i>Primavera Business Software Solutions, S.A.</i>
PDM	<i>Problem Domain Matrix</i>
PIM	<i>Platform-Independent Model</i>
PSM	<i>Platform-Specific Model</i>
QVT	<i>Query/View/Transformation</i>
RUP	<i>Rational Unified Process</i>
SDK	<i>Software Development Kit</i>

SDL	<i>Specification and Description Language</i>
SDM	<i>Solution Domain Matrix</i>
SF	<i>Software Factories</i>
SPL	<i>Software Product Line</i>
SQL	<i>Structured Query Language</i>
UML	<i>Unified Modelling Language</i>
VA	<i>Virtual Automation</i>
VAT	<i>Value Added Tax</i>