ENHANCING AN ATL TRANSFORMATION WITH TRACEABILITY

Laura Felice, Marcela Ridao, Maria Carmen Leonardi and
Maria Virginia Mauco

INTIA, Departamento de Computación y Sistemas
Universidad Nacional del Centro de la Provincia de Buenos Aires
Tandil, Argentina
felice@exa.unicen.edu.ar, mridao@exa.unicen.edu.ar,
cleonard@exa.unicen.edu.ar, vmauco@exa.unicen.edu.ar

ABSTRACT

Model transformation is widely recognized as a key issue in model engineering approaches. In previous work, we have developed an ATL transformation that implements a strategy to obtain a set of Raise Specification Language (RSL) modules from Feature Models (FM). In this paper, we present an improvement to this strategy by defining another complementary and independent model, allowing the incorporation of traceability information to the original transformation. The proposed mechanism allows capturing and representing the relationships created by the application of the transformation rules.

KEYWORDS

Raise Specification Language, Feature Models, Traceability, MDD, ATL.

1. INTRODUCTION

As formal methods offer a wide spectrum of possible paths towards designing high-quality software, in the academia and the industry have adopted them as an alternative of development, especially where safety or security is important [1]. By using formal methods early in the software development process, ambiguities, incompleteness, inconsistencies, errors, or misunderstandings can be detected, avoiding their discovery during costly testing and debugging phases.

However, formal specifications are unfamiliar to stakeholders, whose active participation is crucial in the first stages of software development process to understand and communicate the problem. This holds in Domain Analysis (DA), because its first stage is to capture the knowledge of a particular domain, making necessary to have a model that is comprehensible by software engineers and domain experts. To contribute to bridge the gap between DA and formal specifications, we have been working in the integration of domain analysis phase into the RAISE Formal Method [2]. The main purpose is to specify a family of systems to produce qualitative and reliable applications in a domain, promoting early reuse and reducing development costs. Feature models were used to represent domain analysis because they facilitate the customization of software requirements.

David C. Wyld et al. (Eds) : SAI, CDKP, ICAITA, NeCoM, SEAS, CMCA, ASUC, Signal - 2014
pp. 219–230, 2014. © CS & IT-CSCP 2014 DOI : 10.5121/csit.2014.41121

The integration between the models implies the definition of rules to derive an initial hierarchy of RSL types from a FM. We use the structural information of the FM to derive RSL (Raise Specification Language) [3] constructs following one of the several proposals that facilitate the construction of FM: the Feature-Oriented Reuse Method (FORM) [4]. In order to fit the main proposal of enhancement of formal developments with the RAISE Method into Model Driven Development (MDD) paradigm [5], an ATL (Atlas Transformation Language) [6] transformation has been developed. This transformation allows the automatic derivation of a first abstract RSL specification of a domain starting from a FM. The ATL rules define how features and relationships between them (the source model) are matched and navigated to produce the RSL specification (the target model) [7]. The rules follow closely the principles proposed in the RAISE Method, so this first and still incomplete specification may be later developed into a more concrete one following the RAISE Method steps. The overall strategy is explained in [8]. In this work, we improve this transformation by providing a simple trace mechanism that creates a trace relationship between the elements of the source and target metamodels.

The paper is organized as follows: In Section 2 we introduce the derivation process. The core of the paper is in section 3, were we describe the ATL transformation that obtain RSL schemes from FM with the incorporation of the trace mechanism, and exemplify it with the case study in section 4. Finally, in Section 5 we present some conclusions and outline possible future work.

2. AN OVERVIEW OF THE DERIVATION PROCESS

We have been working in the integration of DA models and formal specifications, giving a strategy to dine a set of RSL schemes from FM. The strategy begins with the analysis of features into the model to arrive to RSL schemes. Then, different constructions of the FM are analyzed in order to complete the structure of the schemes. Relationships between schemes are modelled from the feature information. The result of these steps is a set of schemes that serves as a basis for the RSL scheme hierarchy, reducing the gap between analysis and specification phases. The full derivation process may be found in [9]. In [8] we presented the rules to define schemes in an automatic way. These rules are a simplification of the derivation process with the objective of defining an automatic transformation aligned with Model Driven Architecture (MDA) framework.

The transformation rules are defined by the following mappings:

- A Feature Model is mapped into one or more RSL modules hierarchies.
- A Feature Diagram (FD) is mapped into a RSL modules hierarchy.
- A Concept Feature (root feature) is mapped into a RSL class expression with a type of interest.
- Features that are not concept features are mapped into RSL modules.
- Relationships
    - Grouping is mapped into RSL schemes relations.
    - Dependencies between features are mapped into RSL axioms expressing the corresponding restriction.

Rule 1: Mapping Feature Model

Description: this transformation declares that a FM will be mapped into one or more RSL modules hierarchy. Each hierarchy will be derived from a FD.

Transformation:

- a FM is transformed into one or more RSL schemes

Rule 2: Mapping Feature Diagram

Description: this transformation declares that an initial hierarchy structure of RSL modules will be derived from a FM diagram. The hierarchy could have modifications later when the evolution of the specification is made.

Transformation:

- each feature of the FD will be mapped into schemes according to the following rules.

Rule 3: Concept2Scheme (mapping Concept feature)

Description: this transformation declares that the most abstract RSL module of the hierarchy will be derived from the feature concept.

Transformation:

- each concept in the FM will be a RSL scheme with the same name
- the RSL scheme will have a type of interest whose name represents the system that is being modeled.

Rule 4: Fea2SchSpec (mapping Mandatory feature)

Description: this transformation declares that the schemes in a hierarchy will be derived from the mandatory features.

Transformation:

- a mandatory feature is transformed into a scheme with the same name as the feature and two clauses:

  - the **value** clause with the following properties:

    - isSolitary with TRUE | FALSE value
    - isMandatory with TRUE value
    - isSelected with TRUE value

  - the **extend** clause with the parent scheme name.

Rule 5: Fea2SchSpec (mapping Optional feature)

Description: this transformation declares that the schemes in a hierarchy will be derived form the optional features selected in a configuration time.

Transformation:

- an optional feature is transformed into a scheme with the same name as the feature and two clauses:

- the **value** clause with the following properties:
  - isSolitary with TRUE | FALSE value
  - isMandatory with FALSE value
  - isSelected with TRUE value
- the **extend** clause with the parent scheme name.

Rule 6: Fea2SchSpec (mapping Parameterized feature)

Description: this transformation declares that a parameterized scheme will be derived from a parameterized feature.

Transformation:

- a parameterized feature is transformed into a parameterized scheme with the same name as the feature and the following clauses:

  - the **context** clause with the list of parameters
  - the **object** clause where the type of the parameters is declared
  - the **value** clause with the following properties:
    - isSolitary with TRUE | FALSE value
    - isMandatory with TRUE | FALSE value
    - isSelected with TRUE value

Rule 7: Aggr2Sch (mapping *Aggregate* relationship)

Description: the statement may express both a composition relationship as well as an aggregate relationship. Thus, a RSL specification will have one or more axioms expressing post-conditions ensuring the relation whole/parts.

Transformation:

- if the feature part is atomic and express simple types, it will be transformed into a component in the scheme corresponding to the parent feature. The component is a **record** into the RSL expression. It has the same treatment that the mandatory, optional or parameterized feature.

- If the feature part is not atomic, the part will be transformed into a RSL expression according to the type of the feature that is being modeled, expressed as an embedded object of the expression that contains it.

Rule 8: GroupOR2Sch (mapping Group OR)

Description: this type of grouping is considered like a ´is-a´ relationship among alternative, mandatory or optional features with a parent feature. Thus a RSL hierarchy of modules will be derived from the structure of the set of features.

Transformation:

- the parent feature is modeled as a RSL scheme with at the least one abstract operation. This scheme will define:

  - the **type** clause with the lower and upper values expressing the group cardinality

- the **value** clause with the restriction expressions of the grouping.
- in later refinement of schemes, the consistence restriction will be mapped into a boolean function.

Rule 9: GroupXOR2Sch (mapping Group XOR)

Description: this type of grouping has the same treatment that the OR group.

Transformation:

- the parent feature is modeled as a RSL scheme with at the least one abstract operation. This scheme will define:
  - the **type** clause with the lower and upper values expressing the group cardinality
  - the **value** clause with the restriction expressions of the grouping.
  - in later refinements of schemes, the consistence restriction will be mapped into a boolean function.

Rule 10: FeaReq2SchSpec (Mapping Requires)

Description: this restriction will generate two RSL schemes derived from two features related by the requires relationship.

Transformation:

- the supplier feature is mapped to a RSL scheme with the same name as the supplier feature
- the requester feature is mapped to a RSL scheme with the same name as the requester feature
- an axiom that defines the implication in the requester scheme expressing the requires restriction.

Rule 11: FeaExc2SchSpec (Mapping Excludes)

Description: this restriction will generate a RSL scheme derived from the relations between two features by the exclude relationship.

Transformation:

- the supplier feature is mapped to a RSL scheme with the same name as the supplier feature
- an axiom that defines the implication in the requester scheme expressing the excludes restriction.

3. INCORPORATION OF TRACEABILTIY INTO THE ATL TRANSFORMATION

This section describes the ATL transformation that obtains RSL schemes from FM with the incorporation of the trace mechanism. The complete description of the transformation rules may be found in [9]. This ATL definition represents the rules described before and it is a single module involving several ATL Rules (both matched and lazy rules) along with a set of helpers. In order to define the transformation and execute it, we must define the source and target models as

an Ecore Metamodel. Traceability is also implemented as a separate model, as we describe later in this section.

3.1. Source and Target Models

Figure 1 shows a diagram with the FM Metamodel defined for the ATL transformation. Also, we have defined an Ecore Metamodel for RSL by using a simplified version (Figure 2).

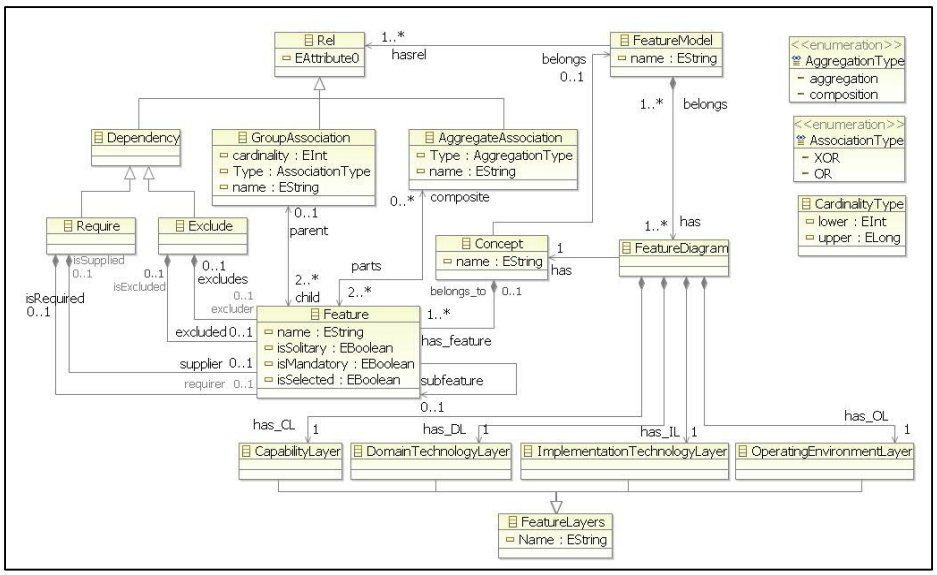

Figure 1. Feature Model Metamodel

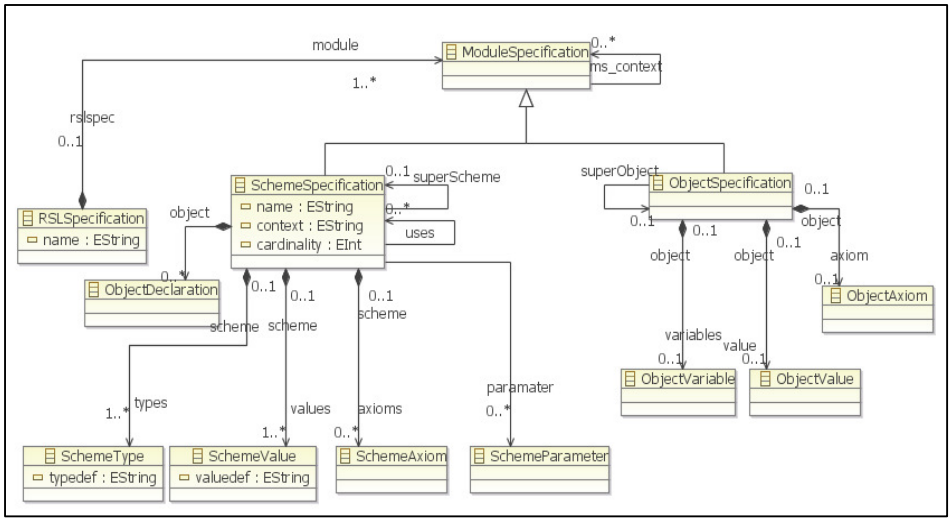

Figure 2. RSL Metamodel

3.2. Traceability information

The transformation process we have proposed allows a simple trace mechanism, based on [10] by creating a trace relationship between the source and the target elements of the corresponding

metamodel according to each transformation rule. For example, each RSL scheme of the module hierarchy is related with one feature because the scheme was originated from one of them. Each of the relationships has their own semantic, and there may be more than one relationship between those components, depending on the rules.

Table 1 shows each trace relationship between elements of the source (FM) and target (RSL) metamodels: the left column represents the source elements and the top row, the target elements. Cells with data indicate a trace relationship. Trace shows the relationships that give rise to new elements in the target metamodel from elements in the source metamodel, i.e. forward relationships, but from them backward traceability ones may be also obtained. The trace relationships are originated from the application a particular transformation rule.

For each trace relationship the following item are described in Table 1:

- Cardinality of source: how many elements were used to create the new element (Table 1, in the left side of the parenthesis),
- Cardinality of target: how many elements are created in that relationship (Table 1, in the right side of the parenthesis),
- Name of the rule that originated the trace relationship.

Table 1. Trace Relationship between FM and RSL generated by the application of the rules

Target / Source	SchemeSpecification	SchemeTypes	ObjectDeclaration	SchemeValue
Concept	(1/1)Rule3:Concept2Scheme	(1/1)Rule3: Concept2Scheme		
Feature mandatory	(1/1)Rule 4:Fea2SchSpec	(1/1)Rule4: Fea2SchSpec		(1/n)Rule4:Fea2SchSpec
Feature optional	(1/1)Rule 5:Fea2SchSpec	(1/1)Rule 5: Fea2SchSpec		(1/n)Rule4: Fea2SchSpec
Feature parameterized	(1/1) Rule 6: Fea2SchSpec	(1/1)Rule 6: Fea2SchSpec		(1/n)Rule4: Fea2SchSpec
Aggregate	(1/n) Rule 7: Aggr2Sch		(1/n) Rule7: Aggr2Sch	
GroupORAssociation	(1/n) Rule8:GroupOR2Sch			
GroupXORAssociation	(1/n) Rule9:GroupXOR2Sch			
Requires relation	(2/1) Rule10:FeaReq2SchSpec			
Excludes relation	(2/1) Rule 11: FeaExc2SchSpec			

3.3. An example of a rule application

In order to exemplify part of the derivation strategy, we describe the ATL transformation corresponding to the Aggregate association (Figure 3) present in a FM Model to RSL schemes

explaining each of the defined rules and helpers. The Transformation Process contains a matched primary rule that guide the overall process of this transformation, the rule aggr2Sch. This rule allows the matching of all features from the FM and defines a RSL scheme for each of them. For each association, the rules identify the name of the association and collect all of the features clustered under this association. These features will be expressed under the objects declaration into a scheme definition. The helper getparts returns the features that are the parts of the aggregate. The lazy rule Feature2Obj has the input features that are part of the aggregate (this condition is implemented by the helper isPartofAggregate), and returns the names of these features. The helper isPartofAggregate verifies the type of association will be an aggregation.

```
module fm2rsl;
create OUT: rsl, trace: Trace from IN: FM;
.......

rule aggr2Sch{
from
       A:FM!AggregateAssociation
to
       S:rsl!SchemeSpecification (
              name <-A.name,
              object <- A.getparts()->collect (feature |
thisModule.Feature2Obj(feature))
              ),
       TL1: Trace!traceLink (
              ruleName <-'aggr2Scheme',
              targetElements <-S)
              do     {
              TL1.refSetValue('sourceElements', A);
         }
}
```

Figure 3. AggregateAssociation Rule

3.4 Implementing the Traceability Mechanism

In order to implement traceability in our transformation process, we adopt the proposal of Jouault [10]. In this work a trace mechanism is defined by considering traceability information as a separate model (Figure 4), and the code to generate trace relationship is added directly to the transformation rules. It is a simple mechanism that supports any form of traceability, and it is used in other transformations.

Following this idea, we have defined our trace metamodel, and added the corresponding code to the rules in order to define the trace relationship presented in Table 1. All the matched and lazy rules are modified in order to define the trace information.

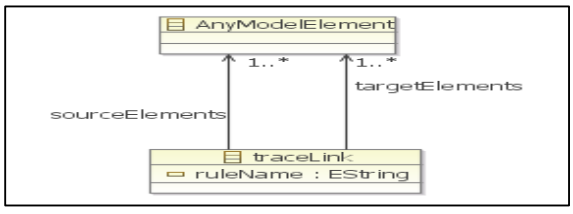

Figure 4: Trace Metamodel

Table 2 shows the implementation of traceability for each transformation shown in Table 1. For our example, during the transformation of a feature to a RSL scheme, trace information is generated for Feature2Object, Feature2ValueIsMandatory, Feature2ValueIsSolitary , Feature2ValueIsSelected lazy rules, besides Aggr2Scheme rule.

Table 2. Implementation of Trace Relationship

Source \ Target	SchemeSpecification	SchemeTypes	ObjectDeclaration	SchemeValue
Concept	ATL rule: Concept2Scheme	ATL lazy rule: Concept2SType		
Feature mandatory	ATL rule: Fea2Scheme	ATL lazy rule: Feature2Type		ATL Lazy rule: Feature2ValueIsMandatory Feature2ValueIs Solitary Feature2ValueIs Selected
Feature optional	ATL rule: Fea2Scheme	ATL lazy rule: Feature2Type		ATL Lazy rule: Feature2ValueIsMandatory Feature2ValueIs Solitary Feature2ValueIs Selected
Feature parameterized	ATL rule: Fea2Scheme	ATL lazy rule: Feature2Type		ATL Lazy rule: Feature2ValueIsMandatory Feature2ValueIs Solitary Feature2ValueIs Selected
Aggregate	ATL rule:Aggr2Scheme		ATL Lazy rule: Feature2Object	ATL Lazy rule: Feature2ValueIsMandatory Feature2ValueIs Solitary Feature2ValueIs Selected
GroupOR Association	ATL rule: group2Scheme	ATL lazy rule: Feature2Type		ATL Lazy rule: Feature2ValueIsMandatory Feature2ValueIs Solitary Feature2ValueIs Selected
GroupXOR Association	ATL rule: group2Scheme	ATL lazy rule: Feature2Type		ATL Lazy rule: Feature2ValueIsMandatory Feature2ValueIs Solitary Feature2ValueIs Selected
Requires relation	ATL rule: Fea2Scheme	ATL lazy rule: Feature2Type		ATL Lazy rule: Feature2ValueIsMandatory Feature2ValueIs Solitary Feature2ValueIs Selected
Excludes relation	ATL rule: Fea2Scheme	ATL lazy rule: Feature2Type		ATL Lazy rule: Feature2ValueIsMandatory Feature2ValueIs Solitary Feature2ValueIs Selected

4. CASE STUDY

We applied the ATL transformation described in this paper to the case study “e-Shop” [12], as it is a well known one. Figures 4 and 5 show some FM features taken from the FM Model of the complete case study, which are necessary to exemplify how the ATL transformation works. For example, we consider the Eshop-Aggregate Association (Figure 4) for the Aggregate Association StoreFront. The features were defined in a XMI format to be used as source in the derivation of the RSL schemes.

Figure 5 shows the XMI definition for the Trace-Eshop-AggregateAssociation, using the Sample Reflective Ecore Model. This Figure presents an extract of the ADITIONAL trace information produced after the ATL transformation. Each trace link includes a reference to an element in the source model (Feature Model), and another one to an element in the target (RAISE Model). For example, trace link with rule name aggr2Scheme produce the trace link for Feature2Obj for the three object declaration (Catalog, BuyPaths and CustomerService) and the trace link for StoreFront.

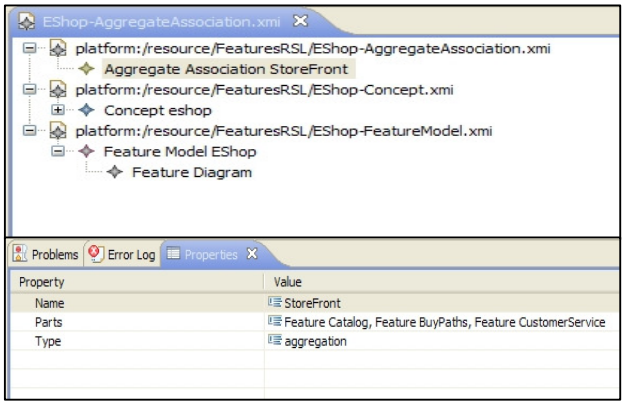

Figure 5. Sample Reflective Ecore Model for Eshop-Aggregate Association

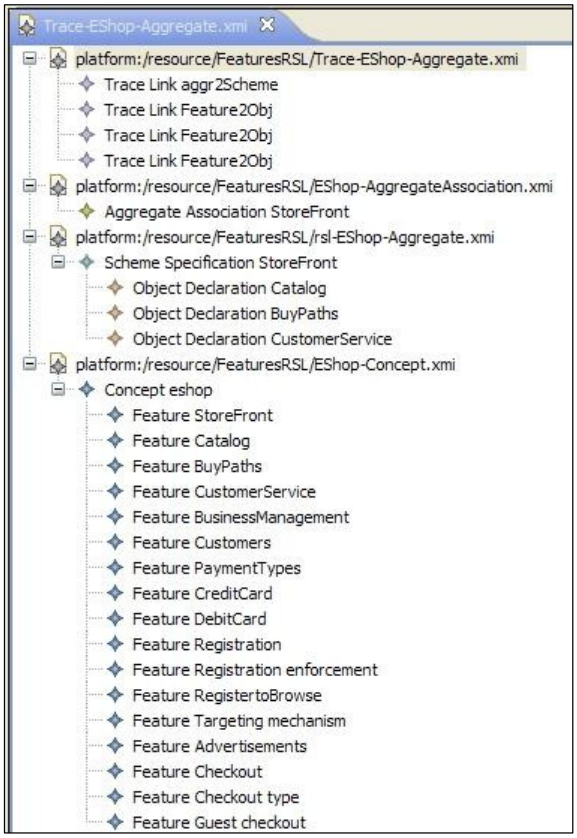

Figure 6. Trace-Eshop-AggregateAssociation

5. CONCLUSIONS AND FUTURE WORK

Traceability plays a crucial role in MDD. The transformation process we have proposed and implemented in the ATL rules allows making a trace between the source and the target models. This concept is quite simple: to follow relationships or links. It is essential for software development because a lot of information is used and produced and it should be kept traceable.

A domain component (source model) is traced forward, for example, when the component is changed and we want to investigate the impact of the change. A scheme is traced backward, for example, when there is a change and we want to understand it, investigating the information used to derive this scheme. Also, we may want to know how a RSL scheme is related with others in a hierarchy, for example, if the derived scheme has a parent or it has some restrictions among features.

In this work, traceability information is easily created but, until the moment it is not managed. There are several difficulties related to the implementation and use of traceability [13]. Wieringa [14] points out that the major problems of realising traceability are organisational, not technical. As future work, we must incorporate trace supporting in order to recorded traces became useful for the entire development process. The traceability generating code is easily added to the ATL code, and can be automated

To help those who are responsible for the specification phase, it is helpful if the work team has traceability policies to traceability information be maintained. Maintaining traceability information is tedious, time-consuming a labour- intensive. Therefore, the policies may be fine, if they cannot be implemented, they are useless [11]. Although in this paper we focused on the transformation rules and the traceability, we know that a lot of information is used and produced and it should be kept related. So, we believe that this approach could be adapted to approach for visualizing traceability in (compositions of) model transformations. The main purpose will be to study the effects of the evolution of a source model, or changes in the transformation model.

REFERENCES

[1] Streitferdt, D., Riebisch, M. & Philippow, I. (2003) "Formal Details of Relations in Feature Models". Proceedings 10th IEEE Symposium and Workshops on Engineering of Computer-Based Systems, pp: 297-304.

[2] George, C; Haxthausen, A; Hughes, S; Milne, R; Prehn, S & Pedersen, JS. (1995) The RAISE Development Method. BCS Practitioner Series, Prentice Hall.

[3] George, C; Haff, P; Havelund, K; Haxthausen, A; Milne, R; Nielsen, CB; Prehn, S. & Wagner, K.R. (1992) The RAISE Specification Language. Prentice Hall.

[4] Kang, K; Kim, S; Lee, J; Kim, K; E. Shin, E; M. & Huh, M. (1998) "FORM: A feature-oriented reuse method with domain-specific reference architectures". Annals of Software Engineering 5, pp 143-168.

[5] Mellor, S., Clark, A. & Futagami, T. (2003) "Model-driven Development". IEEE Software Vol. 20, Nº5.

[6] ATL Transformation Language. Available in: http://www.eclipse.org/atl/.

[7] Felice, Laura; Ridao, Marcela; Mauco, María Virginia & Leonardi, María Carmen. (2011) "Using ATL Transformations to Derive RSL Specifications from Feature Models", Proceedings of the 2011 International Conference on Software Engineering Research & Practice, Volume I, USA, pp: 273 – 279.

[8] Felice, Laura; Ridao, Marcela; Mauco, María Virginia & Leonardi, María Carmen. (2014) "Enhancing Formal Methods with Feature Models in MDD". Encyclopedia of Information Science and Technology, Third Edition. Ed. Mehdi Khosrow-Pour. pp:170-183.

[9] Felice, Laura. (2013) Integration of Domain Analysis Techniques with RSL Specifications. Master Thesis. Facultad de Informática, Universidad Nacional de La Plata, Argentina (http://sedici.unlp.edu.ar/)

[10] Jouault, F. & Kurtev, I. (2005) "Transforming Models with ATL". Proceedings of the Model Transformation in Practice Workshop. MoDELS 2005 Conference.

[11] Kotonya, G & Sommerville, I. (2010) Requirements Engineering. Processes and Techniques. John Wiley & Sons.

[12] Mendonca, M; Branco, M & Cowan, D. (2009) "S.P.L.O.T Software Product Lines Online Tools". In Companion to the 24th ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications. OOSPLA 2009. Florida, USA.

[13] Pinheiro, F. (2004). “Requirement Traceability”. In Perspectives on Software Requirements. Ed. Julio C.S.do Prado Leite, J. Doorn.
[14] Wieringa R.J. (1998). Traceability and Modularity in Software Design. Proceeding of 9th International Workshop in Software Specification and Design. Japan.

Authors

Laura Felice is a computer science assistant professor at the Universidad Nacional del Centro de la Provincia de Buenos Aires from Argentina. She is a member of the Computer Science department. She has a Master´s degree in Software Engineering from Universidad Nacional de La Plata, Argentina. Her main research interests include Software development methodologies, domain engineering and model-driven development. She has been member of the program committee of national and international conferences related to software engineering.
Contact her at:
Computer Science Department, UNCPBA.
Campus Universitario. (7000) Tandil. Buenos Aires. Argentina.
e-mail: lfelice@exa.unicen.edu.ar

Marcela Ridao is a System Engineer and she is a computer science assistant professor at the Universidad Nacional del Centro de la Provincia de Buenos Aires, in Argentina. She has a master's degree in Software Engineering from La Plata University, and she wrote her master’s thesis on Patterns used in Scenario Construction Process. Her research interests include Requirements Engineering, Compilers and real time problems. Currently, she is a doctoral student at La Plata University. She’s writing her doctoral dissertation on Quantitative Techniques Oriented to Semantic Reuse in Requirements Models.
Contact her at:
Computer Science Department, UNCPBA,
Campus Universitario. (7000) Tandil. Buenos Aires. Argentina.
e-mail: mridao@exa.unicen.edu.ar

María Carmen Leonardi is a computer science assistant professor in Universidad Nacional del Centro de la Provincia de Buenos Aires from Argentina. She is member of the Computer Science Department. She has a Master´s degree in Software Engineering from Universidad Nacional de La Plata, Argentina. Her main research interests include software development methodologies, requirements engineering, and model-driven development. She has been member of the program committee of national and international conferences related to software engineering.
Contact her at:
Computer Science Department, UNCPBA,
Campus Universitario. (7000) Tandil. Buenos Aires. Argentina.
e-mail: cleonard@exa.unicen.edu.ar

María Virginia Mauco is a computer science assistant professor in Universidad Nacional del Centro de la Provincia de Buenos Aires from Argentina. She is member of the Computer Science Department. She has a Master´s degree in Software Engineering from Universidad Nacional de La Plata, Argentina. Her main research interests include software development methodologies, requirements engineering, and formal methods. She has been member of the program committee of national and international conferences related to software engineering.
Contact her at:
Computer Science Department, UNCPBA,
Campus Universitario. (7000) Tandil. Buenos Aires. Argentina.
e-mail: vmauco@exa.unicen.edu.ar