# A UML Profile for Feature Diagrams: Initiating a Model Driven Engineering Approach for Software Product Lines

Thibaut Possompès, Christophe Dony, Marianne Huchard, Hervé Rey, Chouki Tibermacine, Xavier Vasques

# A UML profile for feature diagrams: Initiating a model driven engineering approach for software product lines

**Thibaut Possompès, Christophe Dony, Marianne Huchard, Hervé Rey, Chouki Tibermacine, and Xavier Vasques**

*IBM France – PSSC Montpellier*
*Parc Industriel La Pompignane, rue de la Vieille Poste, Montpellier 34006*
*{thibaut.possompes, reyherve, xavier.vasques}@fr.ibm.com*

*LIRMM, CNRS, Université Montpellier 2*
*161, rue Ada, 34095 Montpellier Cedex 5*
*{possompes, dony, huchard, tibermacin}@lirmm.fr*

ABSTRACT. *This paper proposes an instrumented approach to integrate feature diagrams with UML models, via UML profiles and a Rational Software Architect plugin. The concrete contribution is the specification of a new UML profile based upon a meta-model synthesising existing feature diagrams semantics, and a Rational Software Architect (RSA) implementation. Our RSA implementation makes possible to link feature diagrams with UML model artefacts. Indeed, it allows traceability between feature models and other different kinds of models (requirements, class diagrams, sequences or activity diagrams, etc.).*

RÉSUMÉ. *Ce papier propose une approche instrumentée permettant l'intégration de diagrammes de features avec les modèles UML, à l'aide de profils UML et d'un plugin pour Rational Software Architect (RSA). La contribution concrète est la description détaillée d'un nouveau profil UML basé sur un méta-modèle synthétisant la sémantique existante des diagrammes de features, et une implémentation dans RSA. Notre implémentation dans RSA rend possible le lien entre diagrammes de features et éléments UML. En effet, il permet la création de liens de traçabilité entre modèles de features et modèles UML (diagrammes de cas d'utilisation, de classes, séquences, ou activités).*

KEYWORDS: *Feature diagrams, UML profile, software product lines, model driven engineering*

MOTS-CLÉS : *Diagrammes de features, profil UML, lignes de produits logiciels, ingnériérie dirigée par les modèles*

## 1. Introduction

Complex IT projects require efficient tools to support IT analysts, architects and developers when they gather client requirements, domain expert advice and implement software. This is the case in the context of the RIDER project [1] whose purpose is improving energy efficiency of buildings. We think that software product line approach is perfectly appropriate to manage the variations that can be found in our project, and moreover, that feature diagrams will be a great help at gathering specific domains vocabulary and concepts [POS 10].

In this paper, we describe the creation process of a UML 2 profile from a state-of-the-art feature meta-model and of the corresponding Rational Software Architect plug-in. This instrumented approach enables integrating and federating feature diagrams with UML models and artefacts (requirements, class diagrams, sequences or activity diagrams, deployment diagrams). It is intended to be used as a part of a general approach for software product lines and for product generation.

This paper is organized as follows. Section 2 presents the feature meta-model. Section 3 describes how the profile has been derived from the meta-model. Section 4 presents how the profile has been implemented in the modelling tool, validated against industrial concerns and feature diagrams excerpts from our project. Section 5 sums up what has been presented and presents perspectives of further research.

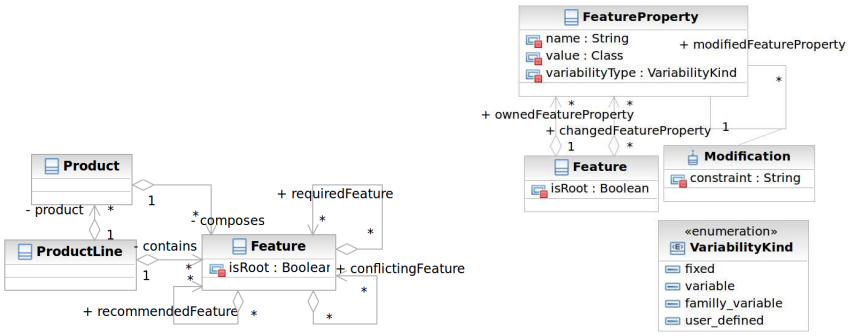
## 2. A feature meta-model

This section describes a meta model synthesising the different interesting points that we previously identified after a state-of-the-art (submitted [POS ]). We chose to transform this meta-model into a UML profile to facilitate the integration into UML models. This work is based upon [ASI 06] which describes a domain ontology for feature models and on the main references on feature models [ZIA 06, GOM 04, ZIA 04, BON 04, RIE 03, KAN 90, KAN 98]. We complete this paper in the way that we produce a meta model that we are using in a very rich industrial project. Our profile implementation is done in Rational Software Architect and we developed a tooling plug-in based upon our UML profile.

### 2.1. *Feature Diagram Meta Model Presentation*

As depicted in Figure 1(a), a product line contains features. A product belongs to one product line and is composed of features; features associated to a product must check some constraints, like mutual exclusion or require relations. Mutual exclusion

1. The RIDER project (“Research for IT as a Driver of EneRgy efficiency”) is led by a consortium of several companies and research laboratories, including IBM and the LIRMM laboratory, interested in improving building energy efficiency by instrumenting it.




(a) Product lines, features and products (b) Feature properties

Figure 1 – Feature meta-model excerpt: Features and their Properties

and require relations are modelled by the *conflict* and *require* relationships. The *recommends* relationship advises the user to choose another feature that could be pertinent.

Feature properties (Figure 1(b)) describe either a feature parameter (*e.g.* the bandwidth capacity of a network) or a property chosen by the user (*e.g.* the frequency of automatic backups of a word processing software). The *VariabilityKind* can be: *fixed*, when the feature property value is fixed throughout all products of the product line; *variable*, when a feature property value can change, within a product, depending on other features properties; *family_variable*, when the feature property could vary from product to product accordingly to the selected features; *user_defined*, when the feature property value can be freely chosen in a given product.

Figure 2 presents the hierarchy relationships and sub-feature groups. A feature and its sub-features are connected by the *RelationshipGroup* class that contains the cardinalities necessary to restrict the number of sub-features to choose. We use special groups like the *OrGroup*, with cardinalities *(0,*)*; *AndGroup*, with cardinalities *(*,*)*; and *XorGroup*, with cardinalities *(0,1)*. A *DirectedBinaryRelationship* links a parent feature and a sub-feature. It can be specialised either by *Enrich*, *Implement*, or *Detail* classes.

In Figure 3, layers and feature sets are linked to the project stakeholders. A stakeholder represents any kind of people; *e.g.* customers, domain experts, IT architect, *etc.*; that can be led to choose features. Feature sets and layers are attached to a specific concern related to the project. A *Layer* represents a view onto the software application. A *FeatureSet* is a *Feature*, which groups features from an arbitrary point of view, *e.g.* for business domain. It also represents the features that must be implemented to fulfil a norm.

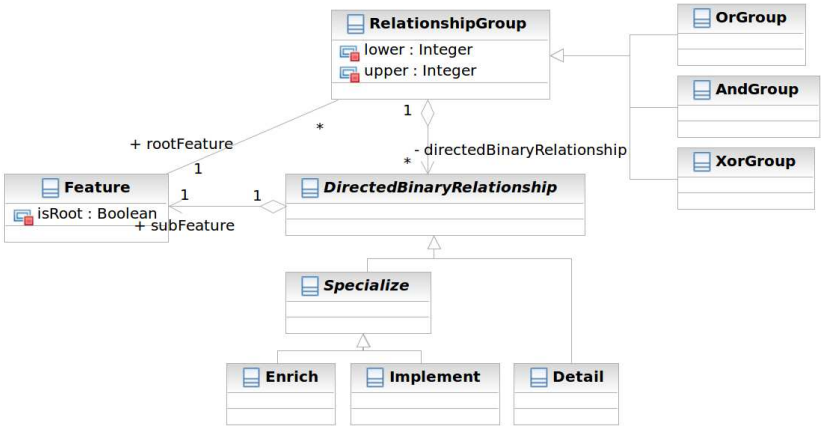



Figure 2 – Feature meta-model excerpt: Groups and hierarchy relationships

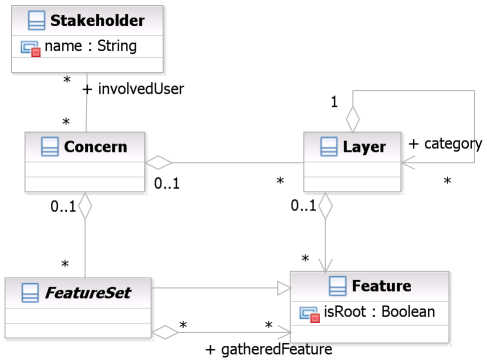



Figure 3 – Feature meta-model excerpt: Layers and feature sets

Figure 4 depicts constraints on feature sets. A feature set can be either; *mutex*, when only one feature can be selected in the feature set; *None*, when there is no constraint between the features composing the feature set; *All*, when all features or none of them can be selected. A *ConstraintRelation* class is a relation between two feature sets. Hence, one feature set can require another one, two feature sets can mutually require each other, or be mutually exclusive.

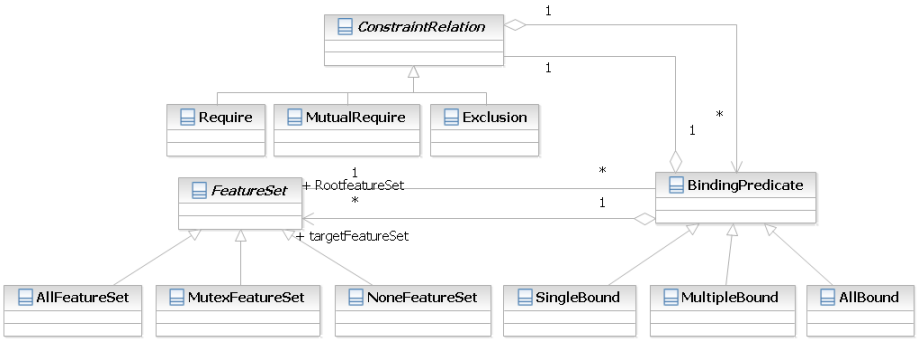



Figure 4 – Feature meta-model excerpt: Constraints on feature sets

### 2.2. *Transformation method*

## 3. Profile model

The meta model describes the syntax of the elements used in a feature model. The profile reuses the concepts described in the meta model and integrates them in UML thanks to the profile semantics. Creating a UML profile consists in creating stereotypes that extend UML meta-classes. Stereotypes are meant to add or subtract semantics from the meta-classes they extend. Hence, the first step to create a profile from a meta-model requires to identify which meta-classes should be transformed into stereotypes, and which UML meta-classes should be extended. This is detailed in the next section. However, the profile could be implemented in very different ways by choosing to extend different UML meta-classes. In our approach, we chose the UML meta-classes that had the closest semantics to our concepts.

### 3.1. *Meta Classes Extensions and Attributes*

Contrary to [CLA 01] we choose to base our feature diagram on *Components*. A component is the UML concept which is the closest to what we want to express because it is a high level view of a software element. This first choice will influence the other UML meta-classes selection. For example, the *port* concept allows us to easily group associations to sub-features. This would be impossible if the *feature* stereotype was extending the *classifier* UML meta-class. Some associations of the meta-model have been implemented as stereotypes.

– *Feature* stereotype extends the meta-class *Component*.

– *Stakeholder* stereotype extends the *Actor* meta-class because they have very close semantics.

– *Concern* extends the *Class* meta-class because we will need class attributes to list layers and feature-set references.

– *ModelRelationship* extends a dependency of a feature to any kind of UML artefact. It extends the *Dependency* meta-class.

– *Layer* extends the *Package* meta-class to ease grouping features into a single place.

– *ProductLine* extends *Package* to centralize all software product lines artefacts in a single place.

– *Product* extends *Component* because it represents a set of features which are considered as a kind of components.

– *FeatureProperty* stereotype extends the *Port* meta-class to allow us representing them as being artefacts directly attached to a feature. They can be linked to other ports or components. The modification of a feature property value can be modelled by textual or OCL constraints placed upon the relationship between two feature properties.

– *RelationshipGroup* extends the *Port* meta-class to represent its belonging to the parent feature. It can be linked only to other features, with directed binary relationships.

– *Modification* extends the *Usage* meta-class.

– *DirectedBinaryRelationship* extends the *Association* meta-class. It is used to link ports to components.

– *BindingPredicate* extends the *Port* meta-class in order to represent the kind of inter feature-set constraint.

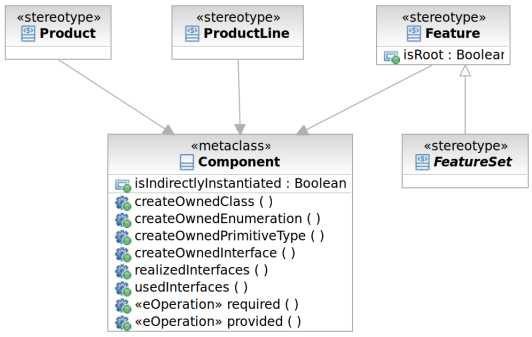
– *ConstraintRelation* extends the *Association* meta-class because it links two feature-sets. It can be navigable or not.

### 3.2. *Resulting Profile*

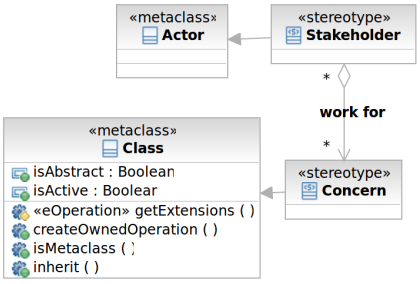
The *ConstraintRelation* stereotype, Figure 6(a), corresponds to the meta-model associations *requiredFeature*, and *conflictingFeature*. It is specialised into respectively the stereotypes *MutualRequire* and *Exclusion*. It is not necessary to link their stereotypes to the *Feature* stereotype because the *Association* and *Component* UML meta-classes are already connected.

The *BindingPredicate*, *SingleBound*, *MultipleBound* and *AllBound* stereotypes, Figures 6(b) and 6(c), extend directly the corresponding meta-class. There is no need to link the binding predicate stereotype to the feature-set stereotype because they respectively extend the *Port* and *Component* UML meta-classes that are already linked in the UML meta-model.

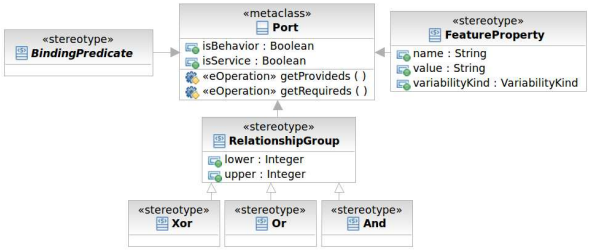
The *FeatureSet* stereotype and its specialisations are directly derived from the corresponding meta-classes.




(a) *Component* UML meta-class extensions

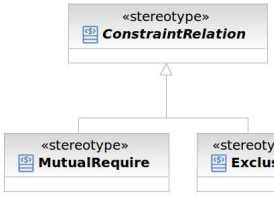



(b) *Class* UML meta-class extensions

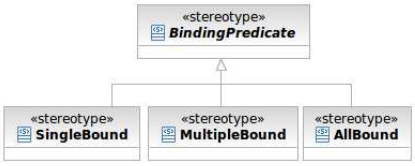



(c) *Port* UML meta-class extensions

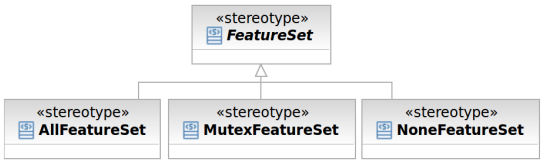
Figure 5 – Meta-classes extensions




(a) Constraint relations between features




(b) Binding predicate of feature sets




(c) Feature sets

Figure 6 – Features and feature sets

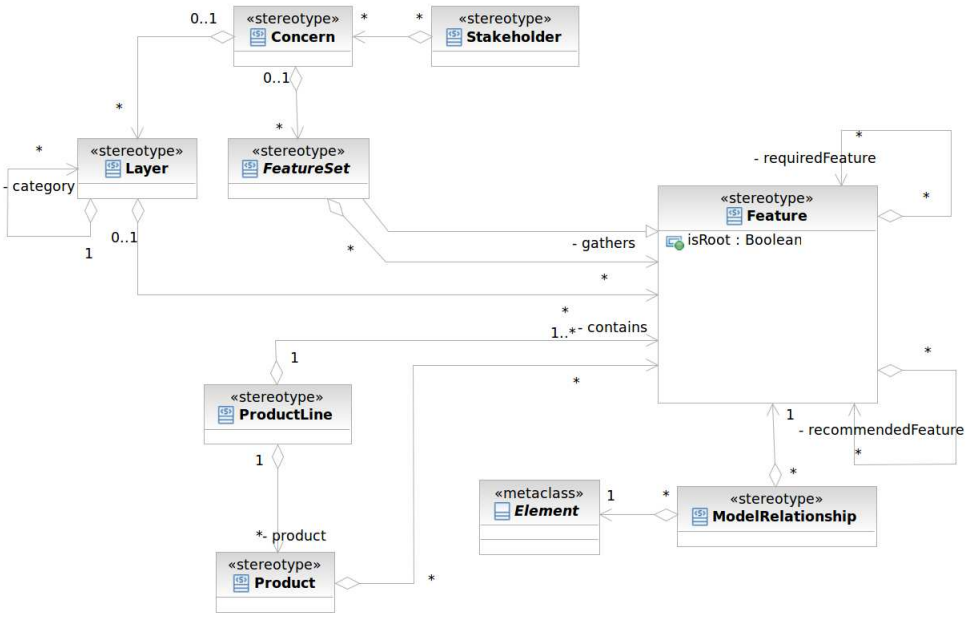



Figure 7 – The *Feature* stereotype

The *requiredFeature* and *recommendedFeature* aggregations have been modelled the same way that in the meta-model because it is more convenient to have attributes in the *feature* stereotype, Figure 7, that list all the required and the recommended features. We could also have modelled these concepts with stereotypes extending the *Association* UML meta-class rather than aggregation attributes.

The *Stakeholder*, *Concern*, *Layer*, *FeatureSet*, *Product* and *ProductLine* stereotypes are derived from the corresponding meta-classes. However the *ModellingElement* meta-class has been derived as the *ModelRelationship* stereotype extending the *Dependency* UML meta-class. It represents a dependency between a feature and any UML element, represented by the *Element* UML meta-class.

The *FeatureProperty* stereotype, Figure 8, is not linked to the feature relationship because it extends the *Port* UML meta-class which is already linked to the *Component* UML meta-class. The *Modification* and *ChangedFeatureProperty* stereotypes extend the *Usage* meta-class in order to show the impact of one feature property on another. The *constraint* attribute describe how the changed feature must be impacted.

The relationships between feature (Figure 9) are directly derived from the meta-model. They are not linked to the *Feature* stereotype because the *Port* and *Association* UML meta-classes are already connected to the *Component* UML meta-class.

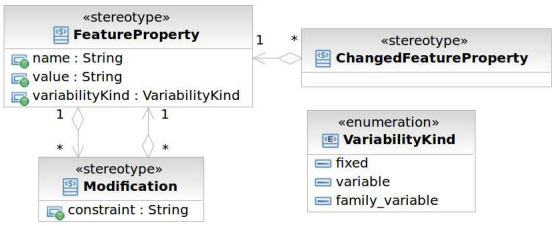



Figure 8 – Feature properties

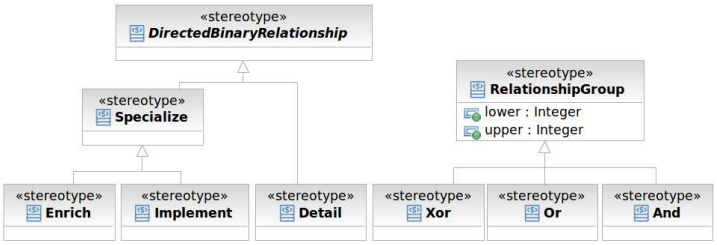



Figure 9 – Feature relationships

## 4. Implementation and Validation

### 4.1. *Common Profile Implementation Tools*

#### 4.1.1. *The Eclipse Modelling Framework and the Graphical Modelling Framework*

The *Eclipse Modelling Framework* (EMF) is a modelling framework integrated in the Eclipse platform, which can be used for building tools and applications based upon a structured data model. It integrates the *ECore* meta-model which is equivalent to the Open Management Group's (OMG) Essential Meta-Object Facility (EMOF). This base framework can be used to support any meta-model. Indeed, it provides an implementation of the OMG Unified Modelling Language (UML) 2.x which can be extended thanks to profiles, the standard UML extension mechanism.

We needed a tool to create feature diagrams that could be integrated in a software development life-cycle using UML. The *Graphical Modelling Framework* (GMF) enables us to develop a graphical editor based upon EMF that fits to our needs.

#### 4.1.2. *Rational Software Architect Profile*

IBM Rational Software Architect is based upon the Rational Modelling Platform which provides a UML modeller, modelling editors, views and tools that are built by using the various services offered by the platform. It also includes several helper components to work with UML models and diagrams. All models managed by the

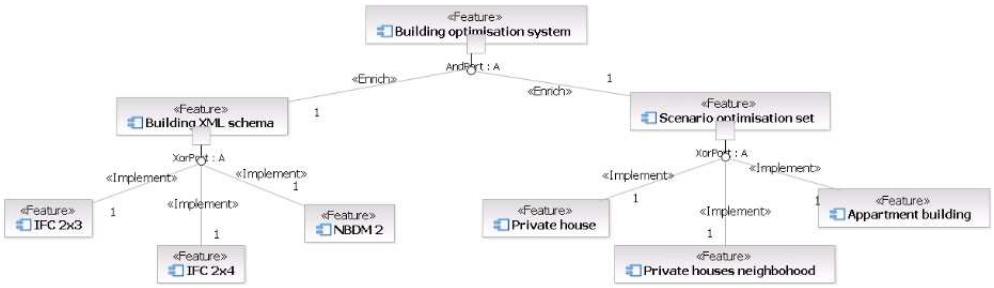



Figure 10 – RIDER feature diagram excerpt

Rational Modelling Platform are instances of EMF models. Hence, using Rational Software Architect allows us to simplify tasks like creating a specific plug-in for integrating feature modelling capabilities into standard EMF-based UML models and diagrams.

We choose to create a Rational Software Architect plug-in instead-of a standard eclipse-based plug-in. Indeed, it facilitates the tooling source code generation by deriving the code from models describing the plug-in structure [MIS 05].

### 4.2. *Examples*

Figure 10 describes the required features of a building management system. The user must choose one building XML schema (to specify the standard describing its building) and a scenario optimisation set accordingly to his building usage.

In Figure 11, the business layer contains the features that describe the energetic optimisations that could be done in a building. All features from the business layer are linked to one or several features from the execution model layer. It specifies which are the modelling standards able to provide the required data to each business feature.

## 5. Conclusion and Perspectives

We have presented how we implemented a feature profile in UML 2. It is based upon a synthesis of existing work that we enhanced to fit the requirements of the project in which this research is applied. We created a Rational Software Architect plug-in to be able in a later time to easily add functionalities required by our industrial project.

We have modelled feature diagrams concerning building modelling standards and IT architecture components. We now need to automate the product generation process and to enhance the user interface of the plug-in.

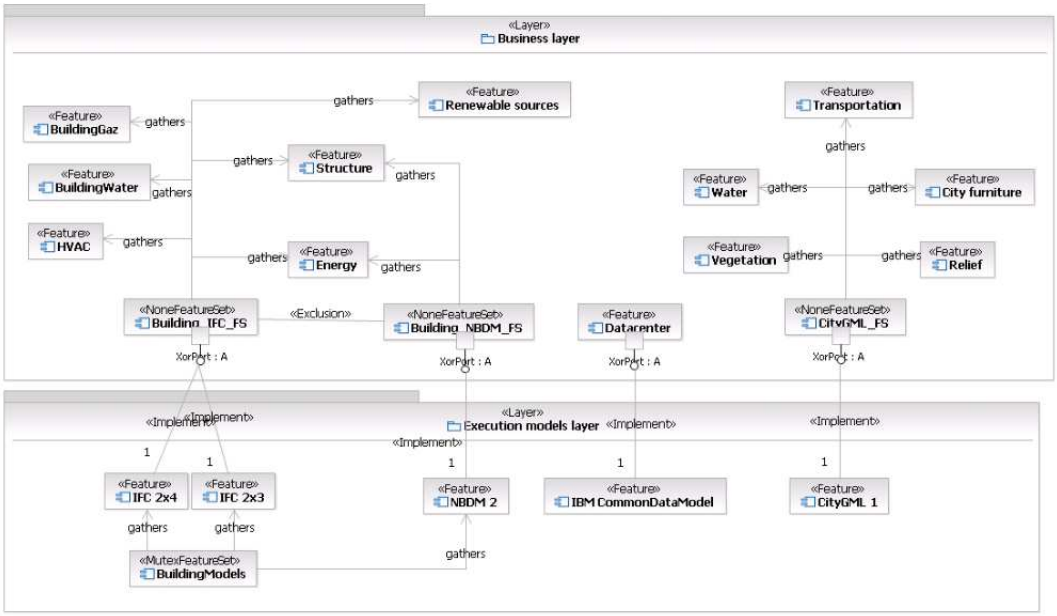



Figure 11 – RIDER feature diagram excerpt

## 6. References

[ASI 06] ASIKAINEN T., MANNISTO T., SOININEN T., "A unified conceptual foundation for feature modelling", *SPLC '06: Proceedings of the 10th International on Software Product Line Conference*, IEEE Computer Society, 2006, p. 31–40.

[BON 04] BONTEMPS Y., HEYMANS P., SCHOBBENS P. Y., TRIGAUX J. C., "Semantics of FODA feature diagrams", *Proceedings SPLC 2004 Workshop on Software Variability Management for Product Derivation–Towards Tool Support*, 2004, p. 48–58.

[CLA 01] CLAUSS M., *Untersuchung der Modellierung von Variabilität in UML*, Technische Universität Dresden, Diplomarbeit, 2001.

[GOM 04] GOMAA H., *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*, Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[KAN 90] KANG K. C., COHEN S. G., HESS J. A., NOVAK W. E., PETERSON A. S., "Feature-Oriented Domain Analysis (FODA) Feasibility Study", report , Nov. 1990, Carnegie-Mellon University Software Engineering Institute.

[KAN 98] KANG K. C., KIM S., LEE J., KIM K., SHIN E., HUH M., "FORM: A feature-oriented reuse method with domain-specific reference architectures", *Annals of Software Engineering*, vol. 5, num. 1, 1998, p. 143–168.

[MIS 05] MISIC D., "Authoring UML profiles using Rational Software Architect and Rational Software Modeler", http://www.ibm.com/developerworks/rational/library/05/0906_dusko/,

Sep. 2005.

[POS ] POSSOMPÈS T., DONY C., HUCHARD M., REY H., TIBERMACINE C., VASQUES X., "Design of a UML profile for feature diagrams and its tooling implementation", *submitted*.

[POS 10] POSSOMPÈS T., DONY C., HUCHARD M., REY H., TIBERMACINE C., VASQUES X., "Towards Software Product Lines Application in the Context of a Smart Building Project", *Proceedings of the 2nd International Workshop on Model-driven Product Line Engineering (MDPLE 2010)*, , 2010.

[RIE 03] RIEBISCH M., "Towards a more precise definition of feature models", *Modelling Variability for Object-Oriented Product Lines*, , 2003, p. 64–76.

[ZIA 04] ZIADI T., HÉLOUËT L., JÉZÉQUEL J. M., "Towards a UML profile for software product lines", , 2004, p. 129–139, Springer Berlin / Heidelberg.

[ZIA 06] ZIADI T., JÉZÉQUEL J.-M., "Product Line Engineering with the UML: Deriving Products", POHL K., Ed., *Software Product Lines*, p. 557-586, Springer Verlag, 2006.