Feature Diagram Technical documentation

Marie Gouyette

Abstract

This documentation presents the technical documentation about feature diagram editor and the tests. This feature diagram editor takes part of the ANR Project Movida.

Published Build date: 17-October-2011
Last modification: $Date:: 2010-05-25 08:39:54#$

CHAPTER

Preface

The feature diagram editor permits to model variability of software architectures. It is divided into two parts, the graphical feature diagram editor himself and a Praxis plugin constraints. This feature diagram editor take into account different notations about feature models and separate features model and base domain models on which it apply.

CHAPTER 1

Choices made on this feature diagram editor

1.1. Feature Diagram Editor

We choose to create a graphical feature diagram editor as an Eclipse plugin. This graphical editor must take into account the following elements : features, decomposition edge such as and, or, xor, card, attributes which can permit to associate metadata on a feature to facilitate the selection into its sons, and a direct mapping with elements from viewpoint model (base model from variability) with addition of model elements from these viewpoint model into the features. These model elements can be any element stored into a model based on EMF (and ecore).

1.2. Constraints plugin

In addition to the feature diagram editor, a plugin constraints was developed in order to help user to create valid feature models. This plugin is written using Praxis rules. Then, we present the rules created on this plugin :

1. noTwoFeaturesHaveSameName : Feature has the same name of another feature in the feature diagram.
2. noParentFeatureAsChildren : The feature children cannot contain their parent feature.
3. noMutexBetweenParentAndChild : There cannot be mutual exclusivity between a feature parent and one of its child.
4. noSeveralMutexOnSameFeature : There cannot have several mutex between the same pair of features.
5. noCyclesOnRequire : For two features f1, if f1 requires f2, f2 cannot requires f1.
6. noBothRequireAndMutexOnSameFeatures : There cannot have both mutex and require constraints between the same pair of features.
7. minCardLargerThanZero : Cardinality minimum on operator card on feature must be equal or larger than zero.
8. noMinGreaterThanMax : Cardinality maximum on operator card on feature must be equal and greater than cardinality minimum.

9. noMaxLessThanMinusOne : Cardinality maximum on operator card on a given feature must be greater and equal than minus one.

10. nbFeaturesMustBeMoreThanMin : The number of children features of the operator Card on the feature must be equal or more than the minimum cardinality.

11. orOperatorMustHaveAtLeastTwoOperands : The or operator on the feature must have at least two features children.

12. alternativeOperatorMustHaveAtLeastTwoOperands : The Xor operator on the feature must have at least two features children.

13. noAncestorFeatureAsChildren : a child feature cannot have one of these ancestors as children.

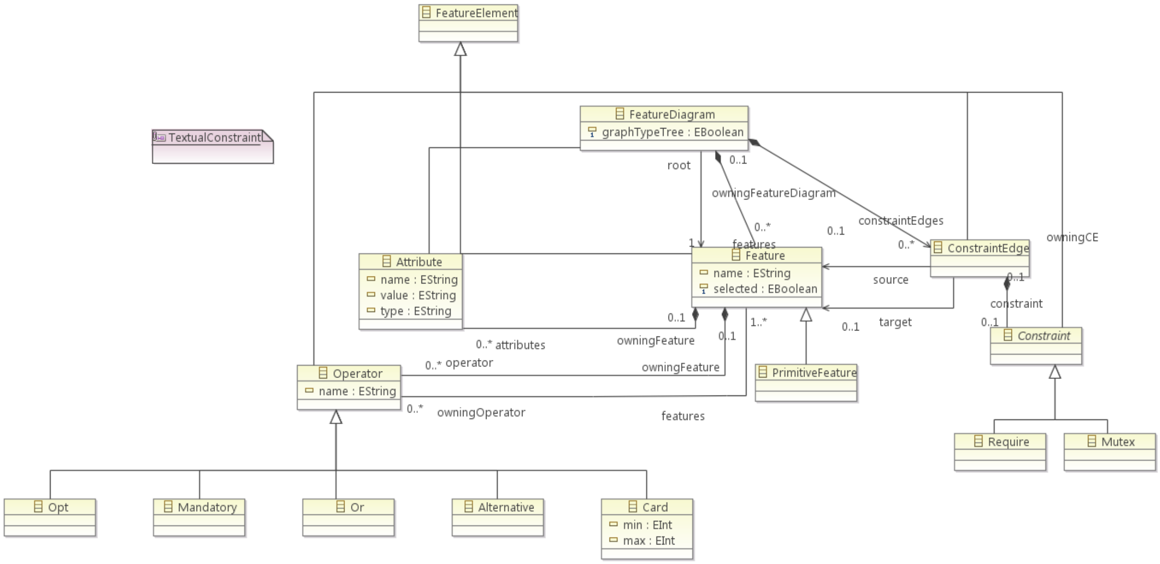
14. noConstraintReflexive : Agiven feature cannot require itself or be mutually exclusive with itself.

15. noMutexBetweenMandatoryFeatureChildren : features with the same feature parent and containec in an and operator cannot be mutually exclusive.

CHAPTER 2

Metamodel used

This section consists in a quick presentation of the metamodel used to create the feature diagram editor. The following picture presents this metamodel.



On this section we will present the feature diagram metamodel used in this tool (cf metamodel above). FeatureDiagram is the root class of the metamodel. This class has an attribute graphTypeTree which permits to determines whether if the feature disgram is a tree feature diagram or a Directed Acyclic Graph (DAG). It also contains a list of features (class Feature) which are represented in the feature diagram as a node. The special root node r is identify by the reference root from FeatureDiagram to Feature. In the metamodel, these operators are subtype of the class Operator, and each feature (class Feature) contains 0 or more operators. The class Feature also contains a list of edges (class Edge) allowing the construction of the setDE of decomposition edges. The set CE of constraint edges is represented in the metamodel by the class ConstraintEdge and they are contained by the class FeatureDiagram. Each ConstraintEdge contains either a Require constraint or a Mutex constraint. Model elements from the base model are stored directly on the Feature metaclass with the modelElements reference. To conclude the Attribute metaclass defines an attribute that we can add on a feature in order to store informations used to determines whether child must be selected. For example, we can add an attribute with the name of a country in a feature and choose to select one of the children feature according to this country.

CHAPTER 3

Limitation of the current version

A future version will permit to not display all parts of the feature diagram at the same time.

CHAPTER 4

Test of feature diagram editor

The tests of the constraints are available on the project **fr.inria.featureDiagramEditor.praxisConstraints.tests**. To retrieve this project go to *File*-> *New* -> *Example* -> *Feature Diagram Editor Sample* -> *Feature Diagram Editor Test Constraints*. Each constraint have one model test stored into a folder with the constraint's name. To test these constraint open the session (file .aird) to see what violates the constraints. If it does not work, select *ViewPoint Selection* and click on *Design*. Then right click on the EMF model -> *Praxis*-> *Activate Consistency Management*. You can see possibles error on the Eclipse Problem view. The file **readMe.txt** on the folder testedConstraints presents each tests.

CHAPTER 5

Conclusion

This document has presented the technical documentation and the tests of the feature diagram editor .