

Présentation de stage TN30

Etudiant : Antoine Toulmé

Responsable pédagogique : Marc Lemercier

■ But

- Cette présentation a pour but de vous présenter une nouvelle approche pour comparer des modèles.

■ Plan

- L'approche orientée modèle
- Les comparateurs existants
- Les bases de la nouvelle approche
- Fonctionnement du moteur de comparaison
- Résultats
- Conclusion

- Définition d'un modèle
 - Une vue de l'esprit analytique et algorithmique, représentant des phénomènes et leurs relations.
- Le modèle informatique
 - Utilisé souvent pour représenter des objets informatiques abstraits (bases de données, applications)
 - De multiples méta-modèles (Merise, UML, Microsoft DSL)

- Le développement orienté modèle
 - Gestion et manipulation de modèles
 - Les modèles deviennent le centre de l'application (MDA)
 - L'OMG encadre le développement orienté modèle avec le MOF, dont une implémentation est EMF.

- Modèles = ressources critiques
 - Modifié souvent et par plusieurs personnes
 - Représentant une application, par exemple
- Outils de développement associés
 - Comparaison de modèles
 - Fusion de modèles
 - Historique de modèles

- Comment comparer deux modèles ?
 - Des éléments
 - Des relations
 - D'appartenance, de contenance
 - D'évènements, etc.

Le conteneur référence l'élément

- Types de différences
 - Ajout d'un élément
 - Suppression d'un élément
 - Modification d'un élément
 - Qui est une somme des différences
 - Ajout d'un attribut
 - Suppression d'un attribut
 - Modification de la valeur d'un attribut

- Lorsque le modèle est ordonné, on peut ajouter une différence de changement de position

- La comparaison par identifiant
 - Elle consiste à comparer les éléments en comparant leurs identifiants.
 - Avantages :
 - Simple
 - Peu ou pas d'erreurs quand on compare deux modèles proches
 - Idéal pour consulter un historique

■ Inconvénients

- La comparaison est éloignée du modèle sémantique
 - Chaque identifiant est unique.
Deux éléments identiques sémantiquement sont vus comme étant différents.
- Les modèles forment un ensemble hermétique.
 - On ne peut comparer ensemble que les modèles issus d'un ancêtre commun.
 - La fusion de modèles est complexe.

- Hypothèse de travail
 - Nous comparons deux éléments d'un modèle en comparant chacune des valeurs de leurs attributs respectifs.
 - Les éléments peuvent être arrangés en couples de façon floue.

- Théorie de l'information
 - $I(m) = \log_2[p(m)]$
 - Plus un attribut apparait, moins il aura de force pour identifier l'élément.
- Refactoring
 - Le refactoring permet de renommer un élément sans changer son comportement.
 - Il faut trouver les différences créées par une opération de refactoring.
 - On crée pour cela l'opération dite de rename.

- Comparaison d'attributs
 - Poids des attributs

$$Poids(A) = \frac{1}{\left[\frac{(\text{nombre d'apparitions})}{(\text{nombre total d'éléments})} \right]} \times \frac{(\text{nombre de valeurs prises})}{(\text{nombre d'apparitions})}$$

- Comparaison d'attributs
 - Notion de distance
 - Entre 0 et 1, 0 signifiant égal.
 - Important pour l'appariement flou
 - Comparaison par type
 - Booléen
 - 0 si les valeurs sont égales, 1 sinon.
 - Chaîne de caractères
 - On utilise l'algorithme de calcul de distance de Levenshtein

■ Algorithme de Levenshtein

- La distance entre deux éléments est le nombre de manipulations à effectuer sur une des deux chaînes pour qu'elle soit égale à l'autre
- Une manipulation peut être un ajout, une suppression, ou une modification de caractère.
- Distance maximale : le max des longueurs des chaînes de caractères.
- On divise le résultat par la distance maximale pour avoir une valeur entre 0 et 1.

- Comparaison de nombres
 - Problème : les nombres ont une sémantique large
 - On les compare comme des chaînes de caractères car ils sont entrés manuellement.

- Comparaison d'éléments
 - La comparaison d'éléments se fait de manière récursive, en utilisant les liens de contenance.
 - On compare les éléments du même niveau
 - On les arrange en couples
 - On passe au niveau suivant en utilisant les enfants de chaque couple.

- Arrangements en couple
 - On compare chacun des éléments du modèle A avec ceux du modèle B
 - Les éléments ayant un taux supérieur a un taux limite sont ajoutés a une liste
 - Les couples de la liste ayant les plus forts taux sont choisis, les couples dont un des éléments est déjà verrouillé sont supprimés.

- **Comparaison a trois**
 - Elle reprend les mêmes éléments, en ajoutant une notion de provenance de la différence.
 - On évalue les cas possibles en essayant de se ramener a une comparaison à deux.
 - Si on ne peut pas, on compare les attributs en utilisant la même approche.

- Le moteur marche et donne des résultats proches de la sémantique du modèle.
- Un jeu de tests montre qu'il est capable de détecter tous les types de différences entre deux modèles.
- Le moteur s'appuie sur une interface Eclipse.

Compare

Differences

- <empty>, <Class> 1, <Class> clazz 88 %
- <empty>, <Class> 2, <Class> classwithaparticulardname 88 %
- <empty>, <Extension>, <deleted> 0 %
- <empty>, <deleted>, <Collaboration> 0 %
- <empty>, <deleted>, <Artifact> 0 %
- <empty>, <deleted>, <Actor> 0 %

left.uml

- <Model> v
 - <Extension>
 - <Class> 1
 - <Class> 2

right.uml

- <Model> v
 - <Class> classwithaparticulardname
 - <Class> clazz
 - <Collaboration>
 - <Artifact>
 - <Actor>
 - <String Expression>
 - <Element Import>
 - <State Machine>

Problems

0 errors, 0 warnings, 0 infos

Description	Resource	Path	Location

The screenshot displays the Intalio software interface for comparing two UML models. The top section, titled "Differences", lists the following items:

- <Class> classwithparticularname, <deleted>, <Class> classwithparticularname 100 %
- <empty>, <Class> 1, <Class> clazz 100 %
 - EObject <Class> clazz added
 - EObject <Class> 1 added
- <Class> clazz, <Class> 2, <deleted> 89 %
- <empty>, <Extension>, <deleted> 0 %
- <empty>, <deleted>, <Collaboration> 0 %
- <empty>, <deleted>, <Artifact> 0 %
- <empty>, <deleted>, <Actor> 0 %
- <empty>, <deleted>, <String Expression> 0 %
- <empty>, <deleted>, <Element Imports> 0 %

The interface shows two model trees side-by-side:

- left.uml:**
 - <Model> v
 - <Class> clazz
 - <Class> classwithparticularname
 - <Collaboration>
- right.uml:**
 - <Model> v
 - <Class> classwithparticularname
 - <Class> clazz
 - <Collaboration>
 - <Artifact>
 - <Actor>
 - <String Expression>
 - <Element Import>
 - <State Machine>

The bottom of the window shows tabs for "Problems", "Navigator", and "Declaration".

- Améliorations et travaux futurs
 - Performances du moteur
 - Ajouter des actions
 - copier un élément
 - Fusion automatique et manuelle
 - Sortie sous forme de texte
 - Intégration à des systèmes de contrôle de version
 - CVS, SVN

- Le développement orienté modèle nécessite plus d'outils
 - Mais ils sont durs à développer
 - Ils doivent avoir des démarches génériques
- Les premières approches de la comparaison de modèles ne vont pas assez loin
 - Basées sur les identifiants
 - Sans rapport avec le domaine sémantique

- Notre approche est payante
 - Elle est assez générique (pas dépendante d'un méta-modèle)
 - Elle respecte la sémantique du modèle
 - Elle arrange les éléments en couple de façon floue

- Cette approche a été implémentée sous la plateforme Eclipse
 - Projet en autonomie
 - Politique d'Intalio pour l'open source
 - Proposé pour rejoindre la plateforme Eclipse