MODEL DRIVEN ENGINEERING

Progetto: *Conflict Resolution*

Marco di Natale, 255660.
Cristina Ciavarro,253188.
Riccardo Iovenitti, matr. 253093

1. Introduzione

Come in tutte le modalità di sviluppo e progettazione di tipo collaborativo, anche la modellazione soffre del noto *problema dei conflitti*.

Tale problematica richiede una particolare gestione dovuta alla natura stessa (tipicamente gerarchica) dei modelli che, a differenza dei normali sorgenti di codice, non permette un confronto linea per linea. La modalità di risoluzione dei conflitti avviene, nel nostro particolare caso, tramite *operazioni di unione*.

2. Il problema del conflitto

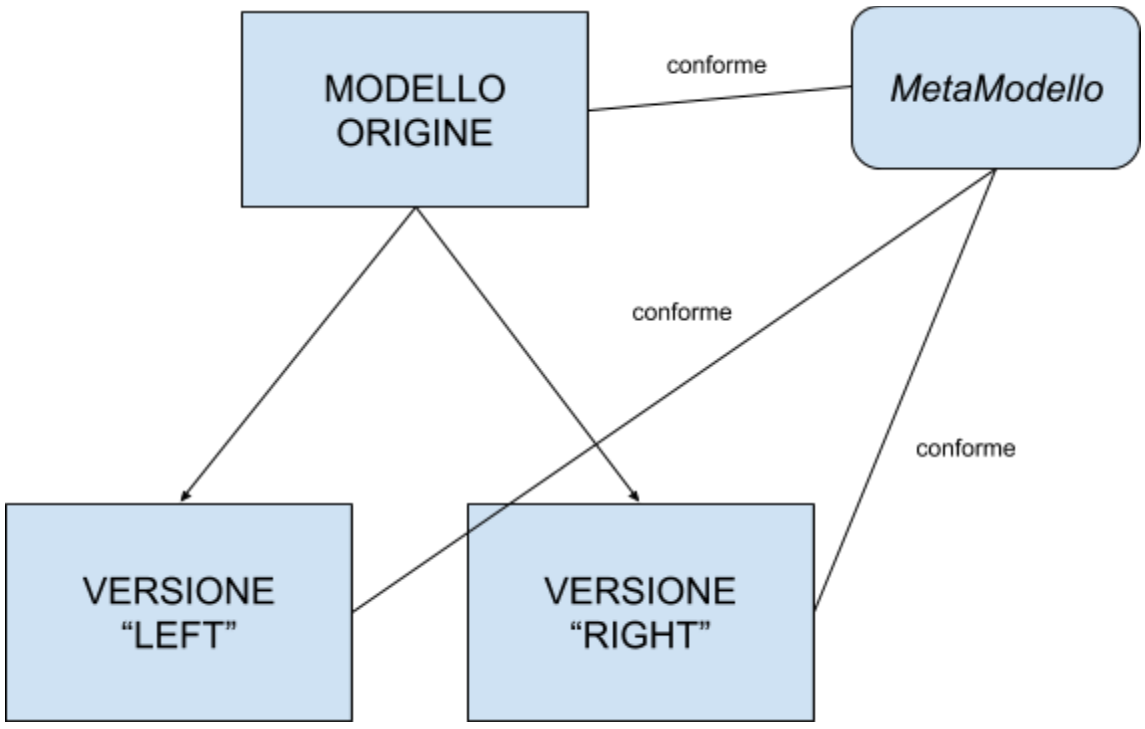
In una tipica situazione di collaborazione si parte da un modello (nel nostro caso si riferisce ai membri e alle relazioni di una famiglia), che può essere modificato da più utenti in maniera contemporanea e indipendente. Di norma i cambiamenti effettuati al modello di partenza presentano delle diversità dovute al particolare approccio risolutivo del modellatore. Il modo più semplice di interagire con i modelli potrebbe essere quello di risolvere immediatamente i conflitti non appena essi vengono generati. Tuttavia, questo modus operandi porta a una perdita inevitabile di informazioni per quanto riguarda le decisioni future: infatti, nel momento in cui ci si accorgesse che era necessaria una diversa risoluzione del conflitto, si dovrebbe necessariamente effettuare un *roll-back* del sistema al un punto immediatamente precedente a quello in cui la decisione, scelta in

precedenza, era stata presa.

Questo problema è quello che la soluzione da noi proposta mira a risolvere, poiché consente di mantenere storia di ogni *incertezza* verificatasi durante il processo di sviluppo e di permettere quindi la sua risoluzione in un secondo momento ed in maniera indipendente.

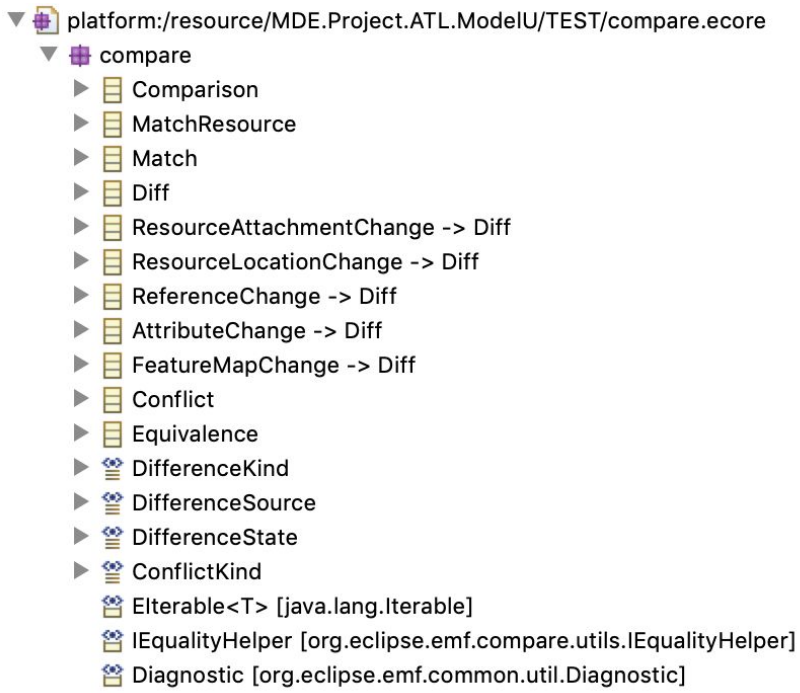
3. Implementazione

Dal punto di vista implementativo è stato utilizzato il tool *EMF Compare* per identificare i conflitti usando l'unione a tre vie su una situazione tipica come indicato in figura:



È possibile determinare le differenze tra due modelli (*Left* e *Right*) tramite l'output di EMF Compare il cui modello di confronto è conforme al

metamodello Ecore (*compare.ecore*).



Utilizzando compare.ecore è possibile estrarre informazioni su ogni differenza trovata confrontando i modelli *Left* e *Right* rispetto al modello di origine. Queste informazioni devono essere quindi salvate in un ulteriore modello contenente le incertezze e che è conforme al metamodello (**Umodel**) - il modello d' incertezza. Umodel è così formato: partendo dal metamodello del sistema preso in considerazione, viene aggiunta, per ogni metaclasse, un'altra metaclasse corrispondente con lo stesso nome ma con una *u* prefissa. Ognuna delle *uMetaclass* conterrà riferimenti ai metadock di origine sinistra e destra coinvolti nel conflitto nonché la tipologia di conflitto rilevata (aggiunte, modifiche o cancellazioni).

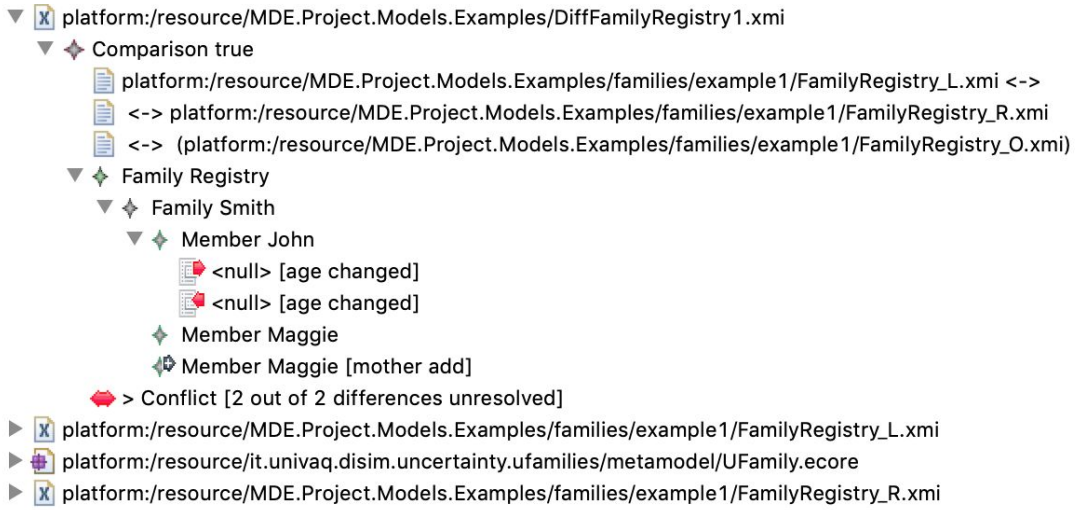
L'obiettivo è quello di creare una trasformazione (in linguaggio *ATL*) che crei un match tra il modello delle differenze e l' Umodel, così da evitare la riscrittura di una specifica trasformazione per ogni tipologia di modello di incertezza. *HOTgenerator*, invece, è una trasformazione di tipo *model-to-text* che consente di prendere un qualsiasi metamodello come

input e di generare il codice ATL illustrato poc’anzi.

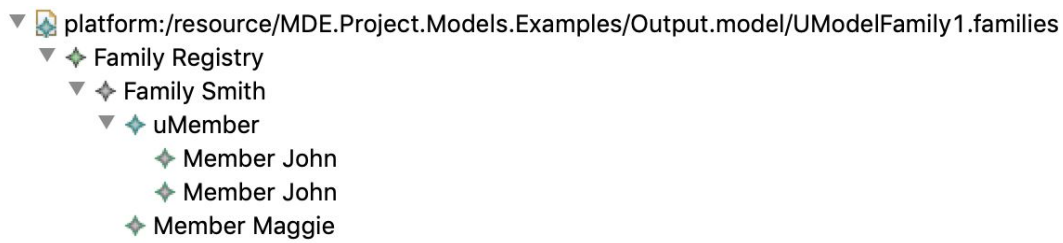
Esempio

Nell'immagine seguente c'è un confronto tra i modelli "Families" generati da EMF Compare, che contiene due conflitti, uno riguardante un cambio di attributo del figlio John e l'altro sul nuovo membro aggiunto. Ognuno di questi conflitti è stato trasformato in 'uMember' conforme al metamodello UFamily. L'esempio appena mostrato è disponibile in Use Case - family - three-way_1 Il progetto ATL compare2UFamily fornisce una trasformazione ad hoc per la metamodella "Families".

EMF Compare



Trasformazione UFamily



Gli esempi sono disponibili nel package *MDE.Project.Models.Examples*. Al suo interno è possibile trovare i seguenti package:

- **families** che contiene gli esempi di modellazione per il modello di prova utilizzato, *Families2.ecore* .

- **university** che contiene gli esempi di modellazione per il modello da noi implementato, *university.ecore*.

- **Output** che contiene tutti gli Umodel generati.

Requisiti

Di seguito riportiamo tutti i plugin necessari all'ambiente Eclipse per permettere la corretta esecuzione del progetto:

- ATL

- Acceleo

- EMF Compare

How to

Per eseguire l'implementazione appena descritta è necessario creare due configurazioni principali: una per Acceleo ed una per ATL.

La prima prevede come input il metamodello UModel

La seconda configurazione prevede come input un metamodello, compare.ecore (modello di input), ed un modello delle differenze in formato XMI conforme a compare.ecore (nel caso del modello Family useremo DiffFamilyRegistry.xmi).
Siccome Acceleo genera l'ATL comprendendo già il link al file

compare.ecore, nella finestra di configurazione troveremo direttamente il link che punta al file compare.ecore, quindi ci limiteremo a inserire manualmente solo il file del modello delle differenze ed a specificare il nome del file di output che vogliamo ci venga restituito, che sarà conforme ad un metamodello UModel.