UNIVERSITÀ DEGLI STUDI DI ROMA
"TOR VERGATA"

Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingegneria Informatica

Corso di Ingegneria del Software 2 - Parte 2

DEFINIZIONE DELLA TRASFORMAZIONE *UML TO LQN* IN LINGUAGGIO ATL

Docenti:
Prof. ANDREA D'AMBROGIO

Studente:
STEFANO PERNA

ANNO ACCADEMICO 2008-2009

Indice

Capitolo 1

Introduzione

Nel settore dei servizi informatici si continua ad assistere alla crescente produzione di applicazioni software orientate allo sfruttamento delle risorse di Internet, che forniscono agli utenti servizi nuovi e sempre più innovativi. Il fenomeno porta alla nascita di applicazioni Web sempre più complesse e sofisticate, in grado di integrare funzionalità diverse, ed è oggetto di investimenti sempre più ingenti da parte di società ed aziende.

In questo contesto, risulta di fondamentale importanza affrontare la progettazione e la costruzione di queste applicazioni avvalendosi dell'ausilio di metodologie, modelli e strumenti di sviluppo idonei, capaci di assicurare livelli qualitativi elevati, mantenendo i costi di sviluppo contenuti.

L?idea alla base dello sviluppo *Model-Driven* è consentire un approccio integrato allo sviluppo del software, in cui la realizzazione di modelli possa essere considerata parte del processo di implementazione. I concetti definiti dal *Model Driven Engeneering* (**MDE**) rappresentano una generalizzazione del *Model Driven Development* (**MDD**) e della *Model Driven Architecture* (**MDA**) definita dall?*Object Management Group* (**OMG**). Quest?ultima, in relazione all'evoluzione dei linguaggi di modellazione del software (**UML**), intende far sì che le applicazioni software vengano generate attraverso un processo di trasformazione di modelli: un approccio aperto e non proprietario alla sfida del cambiamento tecnologico e di business; un?evoluzione del software engineering che fa leva sulla forte e formale separazione tra modelli funzionali (cosa deve fare il sistema) e modelli tecnologici (come usare le tecnologie), costruiti

utilizzando standard non proprietari definiti e mantenuti dall?OMG, da cui possono essere generate le applicazioni eseguibili, virtualmente su qualunque piattaforma, aperta o proprietaria, incluso ma non limitato a Web-Services, .NET, CORBA, Enterprise JavaBeans, C, C++ e Java.

Usando la metodologia **MDA**, la funzionalità del sistema può inizialmente essere definita come un *Platform-independent model* (**PIM**) attraverso un appropriato *Domain Specific Language* (**DSL**) per essere poi trasformata in uno *Platform-specific model* (**PSM**).

La motivazione dietro **MDA** è quella di spostare il carico del lavoro dalla codifica alla modellazione trattando i modelli come i principali artefatti di sviluppo.

Tra i linguaggi DSL usati per risolvere le trasformazioni tra modelli ci sono **QVT** (*Query/View/Transformation*) e **ATL** (*ATLAS Transformation Language*). Entrambi i linguaggi mostrano un?architettura a livelli in cui sono organizzati i loro componenti. Questi linguaggi ci permettono di trasformare un modello *source* (conforme a un metamodello source) in un modello *target* (conforme a un metamodello target). Essendo la trasformazione un modello essa stessa deve essere conforme a sua volta ad un metamodello ATL o QVT.

Il livello di metameta-modeling (MMM o M3) costituisce la base su cui costruire l?architettura di metamodeling, il suo scopo è definire il linguaggio con cui andremo a specificare (istanziare) il metamodello (M2). Il meta-metamodello si trova ai livelli di astrazione più alti rispetto al metamodello ed è tipicamente più compatto rispetto al metamodello che esso descrive. Un MMM può definire più metamodelli e possono esserci più MMM associati a ciascun metamodello. Esempi di meta-metaoggetti contenuti in uno strato di meta-metamodeling sono MetaClass, MetaAttribute e MetaOperation. Il livello di metamodeling (MM o M2) costituisce la base con cui definire lo strato di modeling, definisce il linguaggio che specifica i modelli sottostanti. Un metamodello è una istanza di un MMM ed esempi tipici di metaoggetti in questo livello sono Class, Attribute, Operation, Component. Un modello è una istanza di un metamodello. A questo livello di astrazione (M1) definiamo il linguaggio che descrive il dominio informativo. Esempi di oggetti in questo livello sono StockShare, askPrice, sellLimitOrder, e StockQuoteServer. Per finire, le istanze di un modello sono gli oggetti utente (user data) e rappresentano uno

specifico dominio applicativo. Il *Meta Object Facility* (**MOF**) è il linguaggio per definire metamodelli.

In Figura 1 sono mostrati come sono organizzati i modelli OMG, possiamo notare che MOF risiede in cima alla gerarchia.

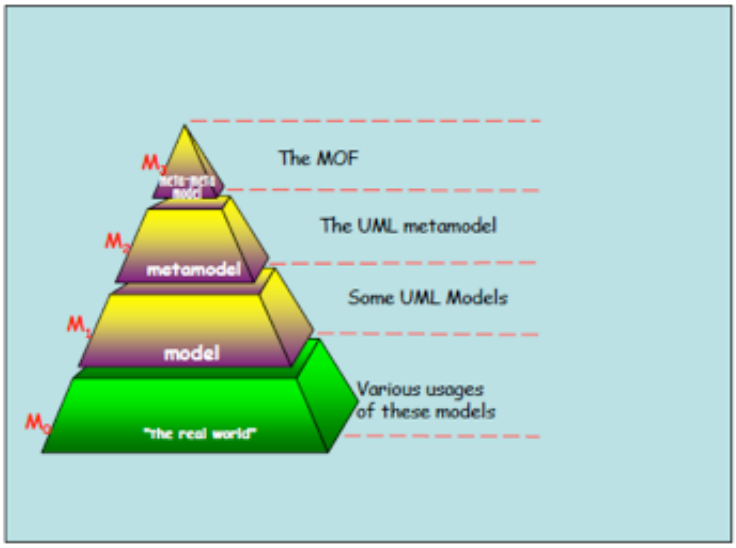

Figura 1.1: Gerarchia dei modelli.

1.1 Obiettivi

In questo lavoro proponiamo una implementazione basata su linguaggio **ATL** della trasformazione di modelli source **UML** (Unified Modeling Language) annotati con il profilo **MARTE** (*Modeling and Analysis of Real-time and Embedded systems*) in modelli target **LQN** (*Layered Queueing Network*), partendo dalla stessa trasformazione definita in linguaggio **QVT** e con modelli source annotati con il profilo **SPT** (*Profile for Schedulability, Performance and Time*).

Capitolo 2

Linguaggi di trasformazioni di modelli

L'obiettivo è quello di trasformare un modello sorgente **Ma** in un modello target **Mb**. Una delle tecniche più comunemente usate per effettuare la trasformazione tra modelli è conosciuta come ?*MetaModel Approach*? basata sul pattern di *Model Transformation* mostrato in Figura 2.1.

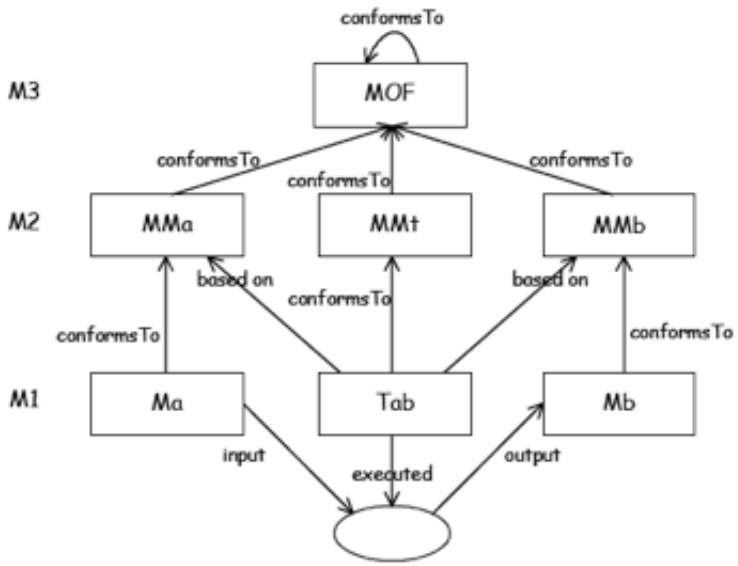

Figura 2.1: Operational contest.

In questo approccio il primo passo consiste nel definire i metamodelli sorgente e target (rispettivamente **MMa** e **MMb**) che forniscono la sintassi astratta per la descrizione dei modelli (rispettivamente **Ma** e **Mb**). Ciascun modello è conforme al proprio metamodello.
Successivamente viene costruita una mappatura tra i due metamodelli (**Tab**), che consiste nello stabilire le corrispondenze tra i costrutti di ogni metamodello: è il programma di trasformazione il cui risultato è la creazione di **Mb** a partire da **Ma**. Questa mappatura può essere definita semplicemente come una tabella in cui sono mostrate le corrispondenze tra i costrutti di ogni metamodello.
In questo caso il linguaggio usato deve essere conforme al suo metamodello **MMt** (ad es. XSL o ATL). Usando un linguaggio eseguibile è possibile effettuare la trasformazione **Tab** da qualsiasi modello di input **Ma** conforme a **MMa** generando il corrispondente modello **Mb** conforme a **MMb**. Nel nostro lavoro **MMa** è il metamodello UML, **MMb** il metamodello LQN, Ma il modello UML di input, **Mb** quello di output conforme a LQN e **MMt** il metamodello ATL. Nei capitoli successivi proviamo a confrontare i due linguaggi per le trasformazioni ATL e QVT evidenziando differenze e aspetti comuni. Questo ci servirà come base nell?implementazione della trasformazione ATL a partire dalla stessa in linguaggio QVT.

2.1 Architettura di QVT

2.2 Architettura di ATL

2.2.1 Regole di trasformazione

2.2.2 Gli Helper

Capitolo 3

Profili di prestazioni

3.1 Il profilo SPT

3.2 Il profilo MARTE

3.3 Conversione tra i profili

Capitolo 4

Il modello LQN

4.1 Motivazioni

Capitolo 5

Trasformazione Uml to Lqn

5.1 Il livello M3

5.2 Il livello M2

5.3 Il livello M1

Capitolo 6

Trasformazione QVT Uml2Lqn

Capitolo 7

Trasformazione ATL Uml2Lqn

Capitolo 8

Conclusioni

Capitolo 9

Bibliografia