Model-Based Development of Safety-Critical Systems

Miguel A. de Miguel

5/6, 2006

Overview

- MDA Objectives and Solutions
- Infrastructures for MDA approaches
- Application of MDA in safe-aware software development
- Safety modeling languages
- Transformations for safety modeling languages
- Open issues

Stakes

- To increase the **industrial competitiveness** in the domain of software systems development
- To **face the growing gap** between the current software development productivity and the increasing demand
- Industrialists need to respond to these ever higher expectations in order to **maintain their first-rate ranking in the development of complex, mission-critical software systems**

Context

- The efficiency and competitiveness of safety critical systems (e.g. air traffic control management) depends on **high-quality software systems**
- There is an **increasing demand from clients** for better, more complex and more reliable software systems while, at the same time, there is a requirement to **reduce development time and costs** in an ever faster evolving technological world
- The complexity of software systems has increased faster than **software development productivity**

Model Driven Development Solutions

- In MDA, **software development** moves from being code- and document-centric to become **model-centric**
- Model is the new basic entity of software development:
  - Automation of the production of most software artifacts such as tests, documentation and code
  - Better capitalization of know-how

Technical Objectives

- Model Based Engineering of Software Products
  - Separation of Concerns at modeling level (domain, technical and platforms)
  - Weaving concerns at platform implementation level
  - Automate model transformation and weaving leveraging generation techniques
- Model designed for change
  - Decoupling all level constraints from their realization and impact
  - Handling variability
- Model projection onto several type of platforms
  - Platform independent model versus platform specific model
  - Generate platform configurations from platform specific models

Overview

- MDA Objectives and Solutions
- Infrastructures for MDA approaches
- Application of MDA in safe-aware software development
- Safety modeling languages
- Transformations for safety modeling languages
- Open issues

MDA Technologies: Modeling Languages and Transformation

- The customization of MDA for specific environments require:
  - **Definition and Customization of modeling languages** for specific modeling purposes
  - **Transformation** facilities that support the automatic transformation of models
- Specialization of Modeling Languages:
  - **Domain**: Defense and Air Traffic Control Systems
  - **Technology**: Component Infrastructures (CCM,EJB), Proprietary platforms
  - **Techniques**: Real-time, Fault-Tolerant, QoS

Support for MDA Approach

Definition of New Modeling Elements

Meta-Models Description

Definition of Meta-Models

Well-Formedness Rules

Concrete Syntax Specification

Support of UML Profiles

Reuse of Profiles

Definition of New Profiles

Generations and Transformations

Query and Transformation Languages

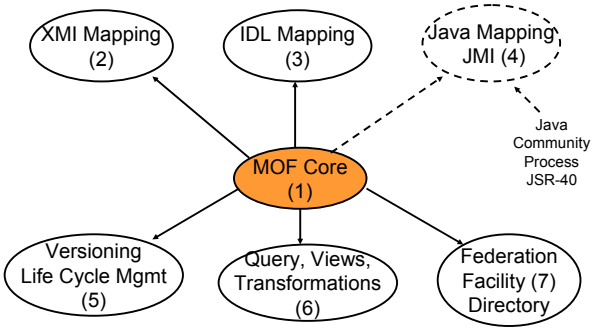
Generation Languages

Traceability

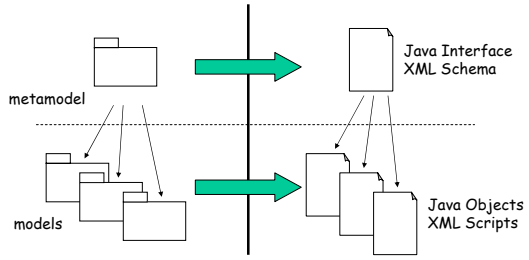
Four Layer MetaModel Architecture

Layer	Description	Example
meta-metamodel	The infrastructure for a metamodeling architecture. Defines the language for specifying metamodels.	*MetaClass, MetaAttribute, MetaOperation*
metamodel	An instance of a metametamodel.Defines the language for specifying a model	*Class, Attribute, Operation, Component*
model	An instance of a metamodel. Defines a language to describe an information domain	*Flight path, Segment of path, Space of Flight, Flight Plan*
user objects (user data)	An instance of a model. Defines a specific information domain.	*<Flight743, replicated>, 654.56, limit_order, <Segment002_Area234>*

MOF 2.0



JMI and XMI Principle



Eclipse: Open Source Modeling Framework

- Eclipse Modeling Framework (EMF)
  - Generation of **java repositories**
    - **Java API, XML persistency**
  - General **edition facilities** (e.g. notification, undo, icons)
  - Generation **model navigation editor**
- UML2
  - **Repository** of **UML2**
  - **Navigation** editor of **UML2**
  - Construction of **Profiles**
- Graphical Modeling Framework (GMF)
  - **Generators if diagram editors and palettes**
  - **Graphical, tooling, and mapping definitions**

Overview

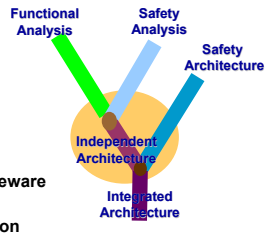
- MDA Objectives and Solutions
- Infrastructures for MDA approaches
- Application of MDA in safe-aware software development
- Safety modeling languages
- Transformations for safety modeling languages
- Open issues

Safe-Aware Modeling Frameworks (SMF)

- Safe-aware modeling language support
  - Application of general modeling languages and **Safety Profiles**
  - **Safety Analysis** Modeling Languages (e.g. FMECA, FTA, ETA)
  - **Safety Architecture** Modeling Languages
- Safe-Aware Transformations
  - **Specification/Implementations of models transformations**
    - From General Languages to Safety Specific Architectures
    - From Safety Specific to Safety Analysis
- Safe-Aware Generation
  - **Specification/Implementation of Generators**
    - **From General Languages + Profile to Safe-Aware Platforms**
    - **Patters for Safety Architectures**

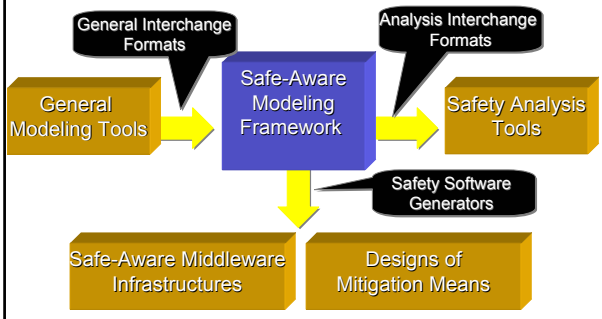
Safe-Aware Engineering Activities

- Functional Analysis
  - **Identification of functionality**
  - **Identification of components**
- Safety Analysis
  - **Risk Assessment: FMECA, FTA**
  - **Assignment of SWAL**
- Safety Infrastructures
  - **Integration of software in middleware and OS infrastructures**
  - **Specific infrastructures/Mitigation Means to reduce the risks**



Integration of different view points in a common modeling language

Model Driven Infrastructures for Safety Software Development



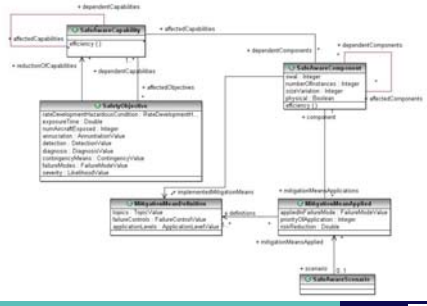
Overview

- MDA Objectives and Solutions
- Infrastructures for MDA approaches
- Application of MDA in safe-aware software development
- Safety modeling languages
- Transformations for safety modeling languages
- Open issues

Safety Concepts in Software Models

- ***Safety Objective***. They are the **safety results** of a requirements phase
- ***Safe-Aware Capability***. They represent software **capabilities** that have **associated safety objectives**
- ***Safe-Aware Component***. It represent logical and physical components that **support safe-aware capabilities**
- ***Mitigation Means Definition***. They define any software **compensation method** that allows avoidance, detection, propagation control or mitigation of effects of failures
- ***Mitigation Means Application***. Components **apply sets of mitigation means** to meet the safety requirements

Abstract Syntax for Safe-Aware Architectures



UML: a Language with Semantic Non-Strict (1/2)

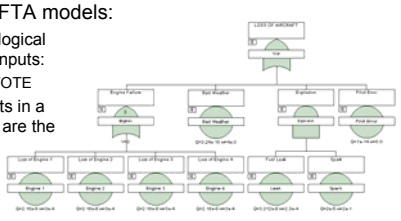
- UML is a **general modeling language** that can be applied in different domains, development phases, or technologies
  - Domain extensions (e.g. UML for the description of Avionic software architectures)
  - Methodological adaptations (e.g. integration of incremental life cycles in modeling driven development)
  - Technical extensions (e.g. modeling real-time applications)
  - Technological extensions (e.g. web server applications construction)
- **Profiles:** UML is Family of Languages
  - Each adaptation requires extend and restrict the UML semantic to the specific domain
  - Most of them are complex concepts that require complex notations for their description

UML: a Language with Semantic Non-Strict (3/2)

- The integration of new concepts in UML requires:
  - Design new notations for the description of new concepts
  - Integrate the new concepts with UML modeling notation
    - Reuse UML modeling elements for the description of new concepts
- In the construction of extension we must:
  - Identify the new concepts to be modeled
  - Design new notations for the description of new concepts
- UML Extension is a solution for the support of MDA in specific platforms, domains and techniques

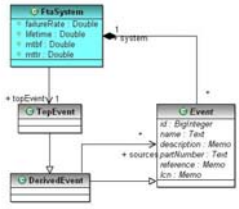
FTA in Eclipse: An example of Safety Analysis Model

- FTA is used during safety assessments to represent the logical interaction and the probabilities of component failures
- Main concepts in FTA models:
  - **Gate**. A gate is a logical function of some inputs:
    - AND, OR and VOTE
  - **Event**. Main events in a fault tree analysis are the **PrimaryEvents**:
    - Failure model



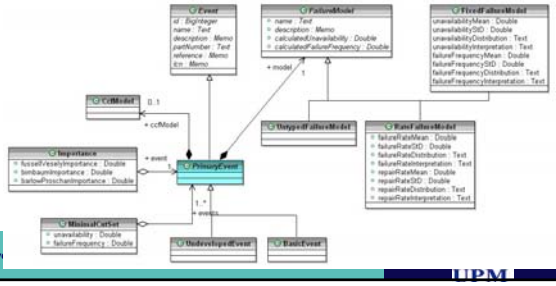
MOF Abstract Syntax of FTA (1/3)

- **FtaSystem** represents a FTA model and it is the composition of a set of events and a top event



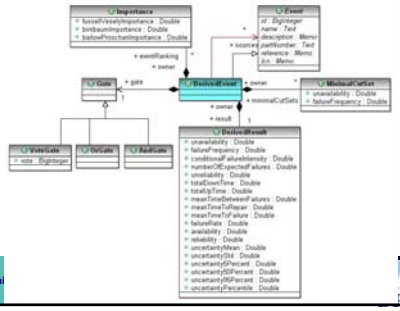
MOF Abstract Syntax of FTA (2/3)

- **Events**. Main events are the **PrimaryEvents** that have an associated failure model. The most typical **RateFailureModel**

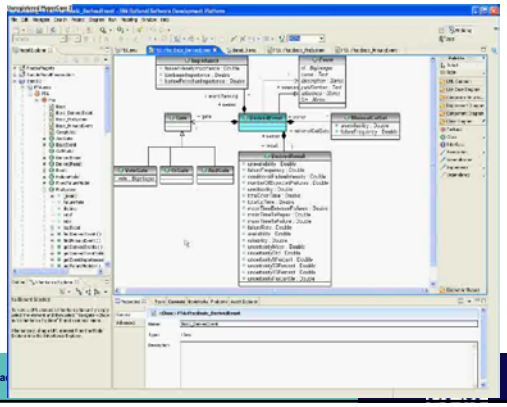


MOF Abstract Syntax of FTA (3/3)

- A **gate** is a logical function of some inputs. The output is an **event derived** from its inputs



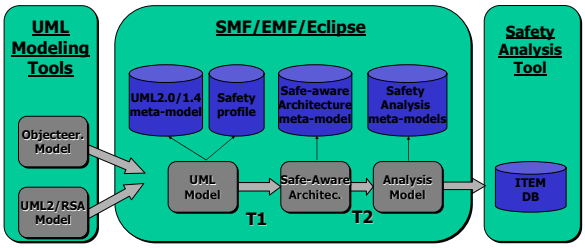
Generation of FTA in EMF



Overview

- MDA Objectives and Solutions
- Infrastructures for MDA approaches
- Application of MDA in safe-aware software development
- Safety modeling languages
- Transformations for safety modeling languages
- Open issues

Transformations in SMF



Design of Model Transformations

- Definition of **transformation rules**
- **Models of rules** based on **queries** and generation **templates** of metamodel roles
- Implementation:
  - QVT or other model transformation languages
  - In Java based on JMI APIs

Overview

- MDA Objectives and Solutions
- Infrastructures for MDA approaches
- Application of MDA in safe-aware software development
- Safety modeling languages
- Transformations for safety modeling languages
- Open issues

General MDA Problems to Be Solved

- MDA standards (e.g. UML, MOF, QVT, JMI) and infrastructures (e.g. Eclipse, EMF, GMF, UML2) are solutions for:
  - Design/Implementation of modeling languages and extensions
  - Construction of model transformators/generators
  - Model Interchange and Java API
- But some problems are not well solved yet:
  - There is not standard solutions to provide modeling services to other tools:
    - E.g. Invocation of remote analysis services
  - There is not *standard* solution to integrate MDA basic technologies (e.g. Metamodels, Profiles, Transformations, JMI code) in assets

Application of MDA technologies in High Critical Systems (1/2)

- **PIM Modeling notations for high critical applications**. Integration in development modeling languages:
  - Real-time concepts (e.g. resource consumption, distributions of responses and source events)
  - Risk assessment (e.g. hazards, compensation means)
  - Quality of software (e.g. quality characteristics, quality supported).
- **High Critical PSM**:
  - PSM of high critical **programming languages**: Ada 2005, RTSJ
  - PSM of high critical **middleware**: RT and FT CORBA

Application of MDA technologies in High Critical Systems (2/2)

- **Integration** of safety analysis in development cycle:
  - **Transformation from PIM and PSM** development models to **analysis** models (e.g. RMA, FTA)
- **Integration** of High Critical PIM and High Critical PSM: Transformation from PIM to PSM
  - From Real-Time PIM to Ada 2005, RTSJ, Real-Time CORBA PSMs
    - Definition of patters in PSM that support PIM concepts (e.g. distributions of source events, scheduling of active resource, protocols of passive resources)
  - Integration of compensation means at PIM and PSM levels: consistency of safety design patterns