Continuous Model-Driven Design / Engineering

Tiziana Margaria

Chair of Service and Software Engineering
Universität Potsdam (Germany)
Edsger Dijkstra: The Humble Programmer

1972

"[The major cause of the software crisis is] that the machines have become several orders of magnitude more powerful!

To put it quite bluntly:

as long as there were no machines, programming was no problem at all;
when we had a few weak computers, programming became a mild problem,
and now we have gigantic computers, programming has become an equally gigantic problem."
ICSE 1996: "Software Crisis: where is it?"

Sir Tony Hoare

"How did Software get so Reliable without Proofs?"

- Defensive programming
- Over engineering
- Requirement verification remains to be key.
Service-oriented Thinking

- Imagine you want to call an elevator
- To reach you, the elevator has to go down
- Which button do you press?
- You do not “drive” the elevator, instead you communicate your intent

Service-orientation: from doing to delegating
from driving to communicating intent
Service Orientation

Participation

Agility

Compliance

User-Centric Process Models
User-Centric Modelling

- User Focus
- User Control
- User Satisfaction

*Driving a car needs no engineer*
In a SO- world

- Programs become Process Models
- Programming becomes Orchestration + Programming in the Small
- Verification becomes more
  - Requirement Validation
  - Property Checking + SLA and
  - Runtime Checking

New Opportunity

Process-Level Formal Methods
Evolving Paradigmas

Component Based Design

M₁  ...  Mₙ

Compilation/Synthesis

Component Library

C₁  ...  Cₙ

Integration

Source Code

a)
More concretely…

XMDD

- Model Library
- Integration as Consistency/Compatibility

'One-Thing'

- Generate Process Code
- Write Sources
- Generate From UML
- Beautify

Genesys

Source Code

- Use Cases
- Business Objects
- Interfaces
- …

System/Application Model

- UML Modeling Tool
- 'Genesys'
- 'One-Thing'
- Full Application

T. Margaria
Continuous Model Driven Engineering
DFKI – 15.7.2009
Main Challenge: Semantics

- Software Engineering
  - static
- Programming languages,
- Semantic Web

Syntactic compliance to the architecture
Main Challenge: Semantics

• Software Engineering  
  – static

• Programming languages, process algebras  
  – operational

• Semantic Web

What happens when one instruction is executed
Main Challenge: Semantics

• Software Engineering
  – static
• Programming languages, process algebras
  – operational
• Semantic Web
  – application domain

What are we talking about
User-Centric Development Cooperation based on One Thing

XMDD
eXtreme Model Driven Design

(MDD + SO + XP + AO(P))

Cooperative development between Business and IT by horizontal and vertical Completion
Horizontal Completion

End-to-End Modelling
- Enterprise and Coalition Spanning
- Push-button Realization
- Immediate Time to Market
- Full Process Control

Includes:
- Documentation
- Use-Cases
- Look & Feel
- Compliance Rules
- Roles & Rights
- ……
Vertical Completion

User-Centric Model Structures
- Hierarchical Process Refinement
- Determination of Business-Roles, -Rights, -Rules
- Service-oriented Integration
- Monitoring

Includes
- Documentation
- Roles and Rights / Permissions
- Domain-specific Constraints
- Test Cases
- Pre/Post Conditions
- Multiple Views
Ongoing Realization in jABC

System/Application Model

UML Modeling Tool
- ENTERPRISE ARCHITECT
- ...

static

✓ Use Cases
✓ Business Objects
✓ Interfaces
✓ ...

dynamic

Business Processes

Generate SLG Code
Write Sources
Generate From UML
Beautify

Full Application
jABC full lifecycle support

Specification
- jABC Core Tracer
- Annotation Ed.
- DBSchema
- FormulaBuilder
- PlanLib

Design Validation
- jABC Core Tracer
- Annotation Ed.
- DBSchema
- LocalChecker
- GEAR
- jETI

Realisation Deployment
- jABC Core Tracer
- jETI
- SIBCreator
- EE Deployer

Test Monitoring
- jITE
- LearnLib
- WebTest

Evolution
- jABC Core Tracer
- LearnLib
- GEAR
- PlanLib
Motivation: jABC

✓ Interpretation & Execution
✓ Analysis & Verification
✓ ... (a lot more!)

www.jabc.de
Many users ....

✓ SOC
✓ CBD
✓ Human
✓ FV

www.jabc.de
Motivation: Model Compilation

jABC Model

generate

Text-based format
Motivation: Model Compilation

jABC Model

generate

Java Servlet

✓ compilable
✓ executable
✓ deployable
Motivation: Model Compilation

jABC Model

generate

✓ executable
✓ deployable
Main Requirements

- jABC Model generates ✓ coincide with modelled behaviour
- jABC Model generates ✓ maximally independent from jABC
Basic Approach

- One generator for each target format!

- Service-oriented: Generators are modelled with jABC!
Compiler bootstrapping
XMDD in

Component Model Library
SIB₁ ⋯ SIBₙ
Macro₁ ⋯ Macron

Feature Library
FLG₁ ⋯ FLGₙ

Heterogeneous Service Models

Integration as Consistency/Compatibility

Global SLG

Compilation/Synthesis

Code Generator

Temporal Constraints and Types
What's specific for Genesys?
Genesys SIBs (Examples)

- Beautify
- BuildENF
- Flush
- LogMessage
- MergeBuffers
- ModelLoader
- OutputTemplate
- StringSetter
- SIBIterator
- SuccessorIterator
Java Class Extruder
Process reusability

Get Package Name \rightarrow Get Class Name \rightarrow Get Root Model

Write Package Name \rightarrow Generate Class Header Top \rightarrow Write Submodel Calls \rightarrow Generate Class Header Bottom

Extrude Models \rightarrow Generate Class Footer \rightarrow MergeBuffers

Get Root Model \rightarrow If is hierarchical model \rightarrow Get Sub Models

Next Sub Model \rightarrow Write Sub Method Call

Servlet Extruder
Property-conform translation

- If a RunVelocity-SIB is used, it must specify a template for Java (special constraint for code generators that produce Java output):

\[(SIB.class == *.RunVelocity) \Rightarrow (SIB.targetLanguage == Java)\]

- A generation profile file before it can be flushed:

Flush precedes

- No more actions should occur after successful or failed generation:

\[(\text{GenerationSuccessful} \lor \text{GenerationFailed}) \Rightarrow AX \neg false\]
ExoMars Rover case study

A Self-Healing Approach for Developing Complex Software Systems
Desired property:
Any step we take is not disastrous. We preserve ReferenceBehavior or we are handling a non-fatal error.

Formally:
\[
[] \ (\text{ReferenceBehavior} \lor \text{ErrorHandling})
\]
Games: For experts
(Formula-oriented view)

\[ X \{ (\text{ReferenceBehavior} \lor \text{ErrorHandling}) \} \]

Diagram showing state transitions and conditions related to reference behavior and error handling.
Illustrates connection between property, model, and proof

Hints at the problem: Drill's release is neither ReferenceBehavior nor ErrorHandling
Case not covered by the specification

Ongoing: Voyager, ANTS, .... ASSL DSL
Enterprise Mashups
Automatic service import
Healthcare: elektronische Fallakte

eFA Prozess
Unterstützung
Einweisung
Enterprise Mashups for Product Lifecycle Management

Parlay-X services (NGN IMS)
Business rules Compliance (GEAR)
Product Lifecycle Management
“15 years experience in a field that is 7 years old”
Services...

for increased productivity

and comfort

(Peter Naur,
NATO Software Engineering Conference
Garmisch Partenkirchen, 1968)
Continuous Model-Driven Design / Engineering

Tiziana Margaria

Chair of Service and Software Engineering
Universität Potsdam (Germany)
margaria@cs.uni-potsdam.de