Extending an ORDBMS: The StateMachine Module

Wolfgang Mahnke\(^1\) Christian Mathis\(^1\) Hans-Peter Steiert\(^2\)

\(^1\)University of Kaiserslautern
Kaiserslautern, Germany

\(^2\)DaimlerChrysler AG
Research & Technology
Ulm, Germany
Generating Software Engineering Repositories using UML

SERUM Framework

- Base Infrastructure
- Model Guidelines
- Design Guidelines

SERUM Design Template

- Configuration Data
- Configuration Rules

Components
- Code-Templates
- Generation Rules
- Aggregation Rules

UML-Repository

Generator

Data Management Service (DMS)

- ORDBMS

refine, adapt & enhance model of DMS

configure DMS

? select model of predesigned DMS

read specification

create DMS

reuse existing artifacts
SERUM: Mapping the Dynamic Model

UML-Repository

StateMachine specification

StateMachine specification

database extension module

database schema

State1

State2

State3

State4

State5

State6

event1

event2

event3

event4

event5

event6

entry/Action2

entry/Action1

entry/Action

action1

action2

action3

action4

action5

action6

Product Data

Meta Data (UML Metamodel)

StateMachine Module

Meta Data (UML Metamodel)

ORDBMS

Data Management Service (DMS)
Goals of the StateMachine Module

• Handling dynamic aspects inside an ORDBMS
  – Observing dynamic integrity constraints
  – Implementing active behavior
  – [Manage workflow activities (needs activity diagrams)]

• Using UML Statecharts as adequate specification for dynamic aspects
  – Provide high-level development environment for UDRs

• Using a generic module (instead of generated modules)
  – Allow dynamic changes of the specification
  – Speed up development cycle
Accessing product data
Sending events
Accessing meta data and internal data
Evaluating conditions and executing actions
The Product Data Schema

Implementation rules:
• each UML class is represented by a ROW TYPE and a typed table
• each UML operation is represented by a user-defined routine (UDR)

Events are generated
• by SQL-trigger for
  – insert events,
  – update events, and
  – delete events
on tables storing UML objects, or
• by code inside the UDR bodies, i.e.
  – a before event and
  – a after event.
The StateMachine Module in Detail

**StateMachine Module**

- **Event Interface**
  - receive all events
  - deliver event to SMM’s global input queue
  - trigger Animation

- **Animation**
  - for each event in global input queue
    - search for receiving SM and put event in local input queue
    - while exists SM with non-empty local input queue animate SM
      - read event from local input queue
      - check if event triggers state transition
      - evaluate fire conditions for triggered state transitions
      - execute exit, transition & entry actions
      - store new state in StateManagement

- **StateMachine-Management**
  - SMM manages internal queues for state machines

- **StateManagement**
  - manages current state of state machines (SM)

- **ModelManagement**
  - stores the state machine specifications (UML metamodel)

- **ConditionEvaluation**
  - evaluates conditions on product data using the ORDBMS’s query facilities

- **ActionExecution**
  - executes actions on product data using the ORDBMS’s query facilities

---

**Event Interface**
- receive all events
- deliver event to SMM’s global input queue
- trigger Animation

**Animation**
- for each event in global input queue
  - search for receiving SM and put event in local input queue
  - while exists SM with non-empty local input queue animate SM
    - read event from local input queue
    - check if event triggers state transition
    - evaluate fire conditions for triggered state transitions
    - execute exit, transition & entry actions
    - store new state in StateManagement

**StateMachine-Management**
- SMM manages internal queues for state machines

**StateManagement**
- manages current state of state machines (SM)

**ModelManagement**
- stores the state machine specifications (UML metamodel)