33. Meta-CASE Toolkits for the Development of Domain-Specific Languages (DSL) and their Editors

Prof. Dr. Uwe Aßmann

Technische Universität Dresden

Institut für Software- und Multimediatechnik

http://st.inf.tu-dresden.de

Version 15-0.3, 16.01.16

Obligatory Reading

- MetaCase. Domain-Specif ic Modeling With Metaedit+: 10 Times Faster Than UML. White paper. http://www.metacase.com/papers/Domain-specif ic_modeling_10X_faster_than_UML.pdf
- MetaCase. Abc To Metacase Technology. http://www.metacase.com/papers/ABC_to_metaCASE.pdf
- Alexander Dotor. Creating a Mancala-Game with Fujaba. Fujaba-Tutorial. Lehrstuhl fu r Angewandte Informatik I. Universität Bayreuth, 2006

Literatur

- [Nill] C. Nill. Analysis and Design Modeling Using Metaphorical Modeling Entities. A Modeling Language for the Tools and Materials Approach. Diplomarbeit Technische Universität Dresden, 2006.
- http://www.metacase.com/support/45/manuals/index.html
- A Comparison of ATL and Story-Driven Modeling (Fujaba-style GRS)
    - http://www.es.tu-darmstadt.de/f leadmin/download/publications/spatzina/PP_AGTIVE_2011.pdf

33.1 Meta-CASE Toolkits

Meta-CASE Toolkits, MDSD toolkits in ONE Technical Space

- A **Meta-CASE-Toolkit** is a metamodel-driven IDE for computer-aided software engineering, for development of IDE and MDSD applications, in *one technical space based on one metalanguage*
  - A software factory should contain several Meta-CASE toolkits
  - Metamodels in the metalanguage are used to control all work:
    - Typing of repositories
    - Generation of repositories with import- and export tools for exchange formats
    - Generation of Editors, typecheckers, visitors, composition tools for models (tools and materials)
  - Modelling of textual and graphic languages
  - Modelling of domain-specific languages and their tools (DSL)

Productivity by Meta-CASE

- Meta-CASE toolkits improve the productivity of a software development team
  - of a team of domain engineers
  - Domain-specific methods are 5 to 10 times faster than using (UML-)notation
  - Reference: Domain-Specific Modeling: 10 Times Faster Than UML; Whitepaper MetaCase 2005; http://www.metacase.com/de/
- Meta-CASE are the most productive tools we know for the construction
  - of DSL
  - of tools
  - of composition systems
  - of IDE (SEU)
- You take part in a course which presents the most productive tools we know!

Examples for Meta-CASE Toolkits

- **MetaEdit+** (commercial): Parameterizable Meta-CASE-Toolkit with
  - Editor for role-oriented metamodels in GOPPR as role-oriented metalanguage
  - Engineering of GUI with Screen-Flow-Language
- AdoXX (commercial), BOC Vienna
- KOGGE, JKOGGE: Generator for graphic IDE
  - KOGGE based on a formal specification and interpreter (Prof. Ebert, Uni Koblenz)
    - http://www.uni-koblenz-landau.de/koblenz/fb4/institute/IST/AGEbert/MainResearch
- Eclipse Modeling Facility (EMOF)
- Netbeans: IDE based on MOF
- MOFLON: IDE based on MOF, with Storyboards (GRS), Logic (OCL) and TGG (GRS)
- Fujaba: with Storyboards (GRS)

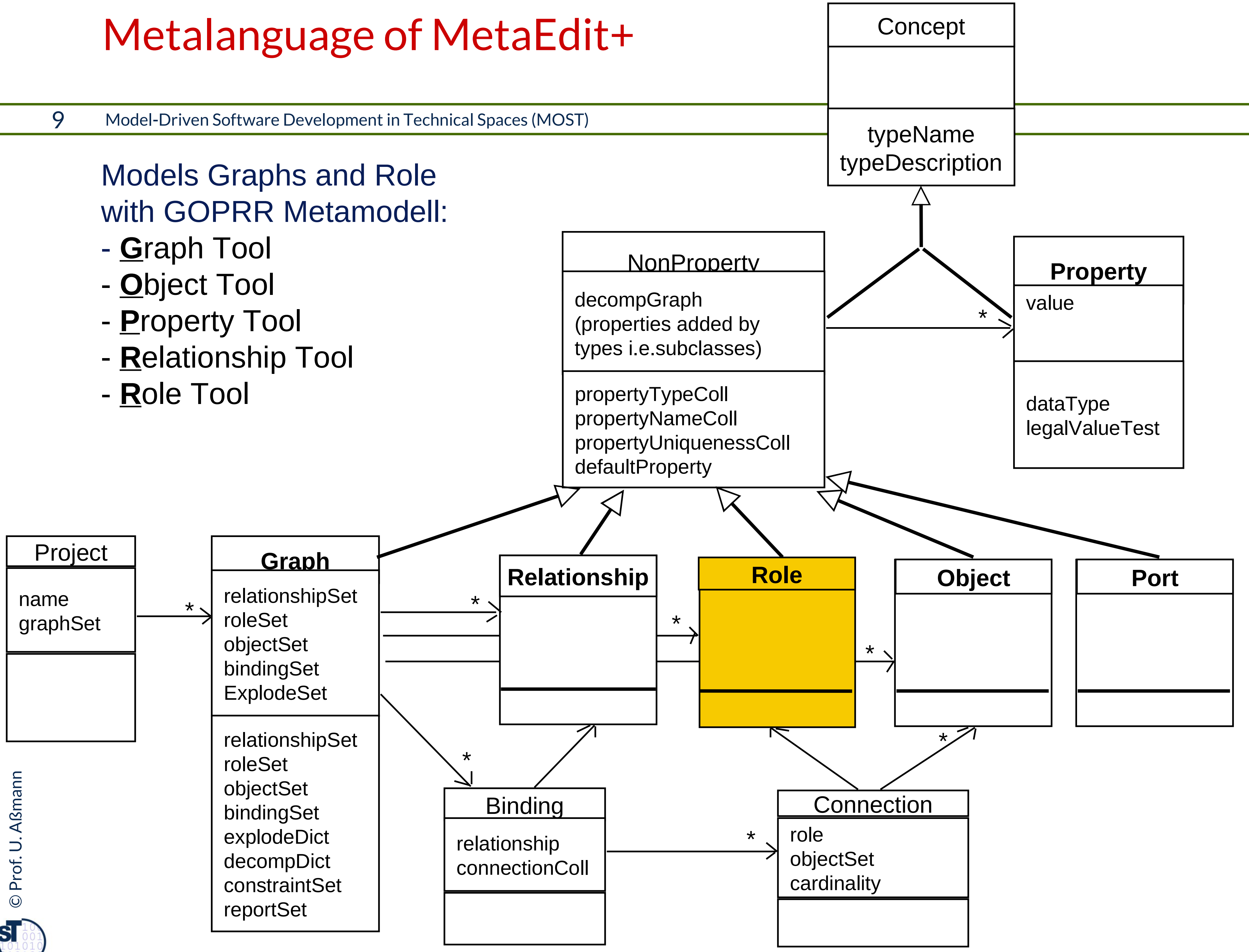
33.2 MetaEdit+ of MetaCase

- A commercial Meta-CASE toolkit
- http://www.metacase.com/download/ Evaluation version
- http://www.metacase.com/cases/dsm_examples.html Many more DSL examples
- http://www.metacase.com/resources.html Articles and handbooks

DRESDEN concept Exzellenz aus Wissenschaft und Kultur

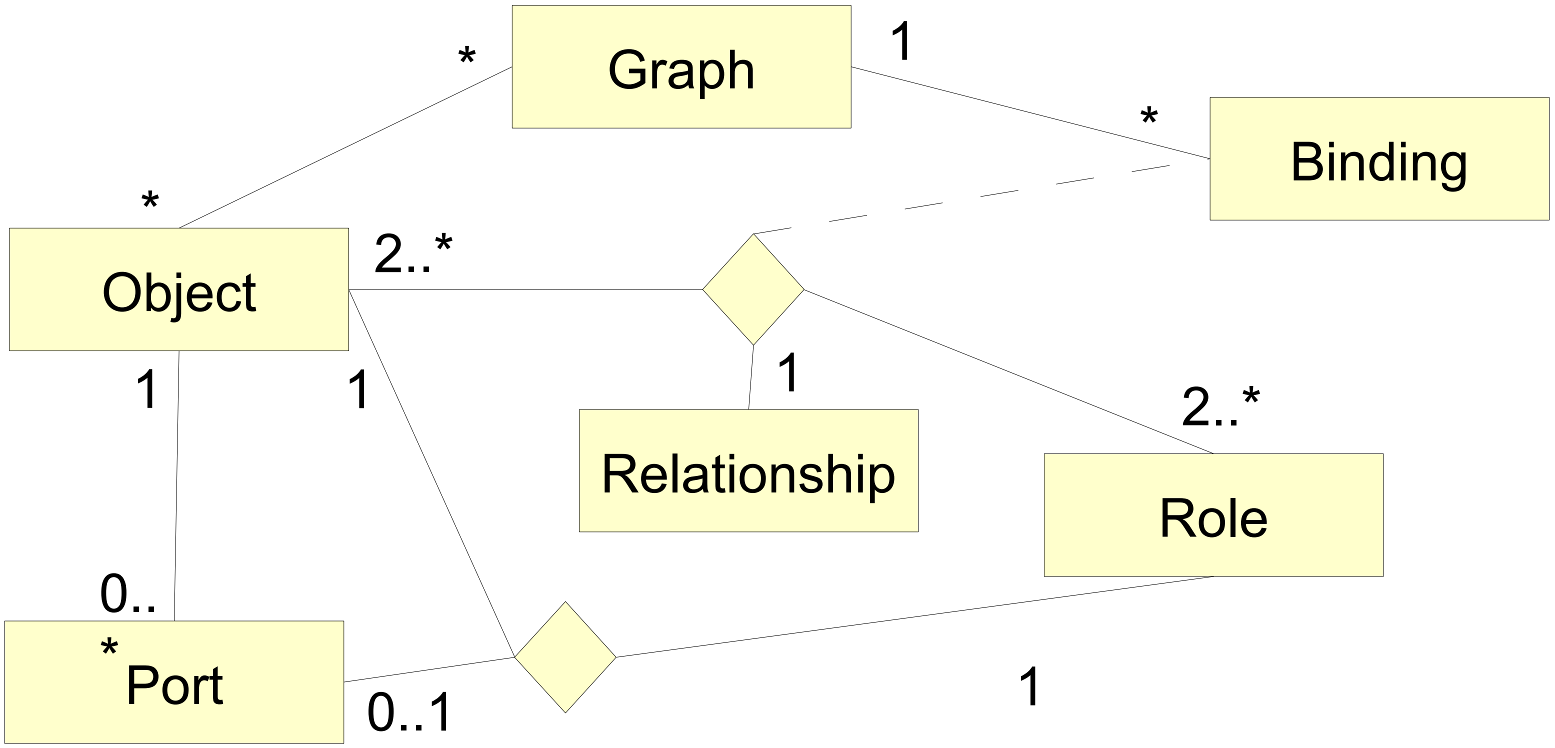
Metalanguage of MetaEdit+

Models Graphs and Role
with GOPRR Metamodell:
- Graph Tool
- Object Tool
- Property Tool
- Relationship Tool
- Role Tool



Wdh: Graph Types in MetaEdit+

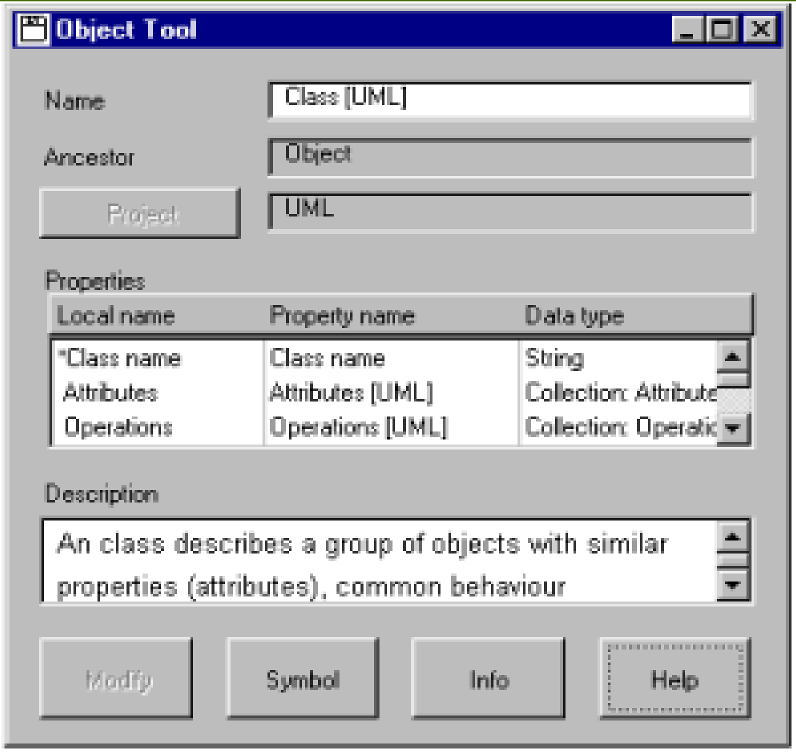
- A **graph type (diagram)** defines:
  - Objects
  - Roles
  - Relationships
  - Allowed Bindings between all entities:
    - a binding consists of a relationship with roles and playing objects



Development of a CASE-Tools with MetaEdit+

Development of language

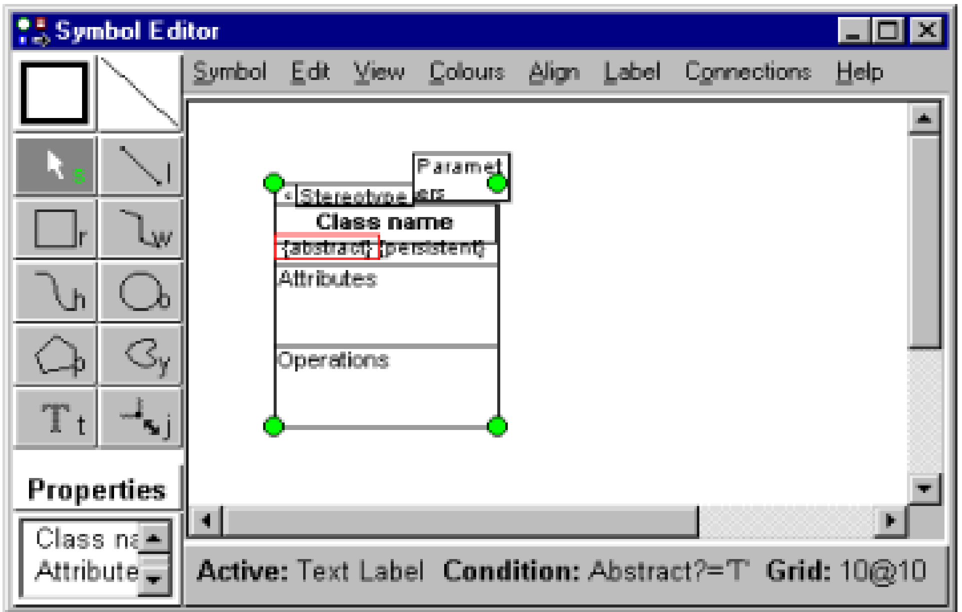
Use of developed language



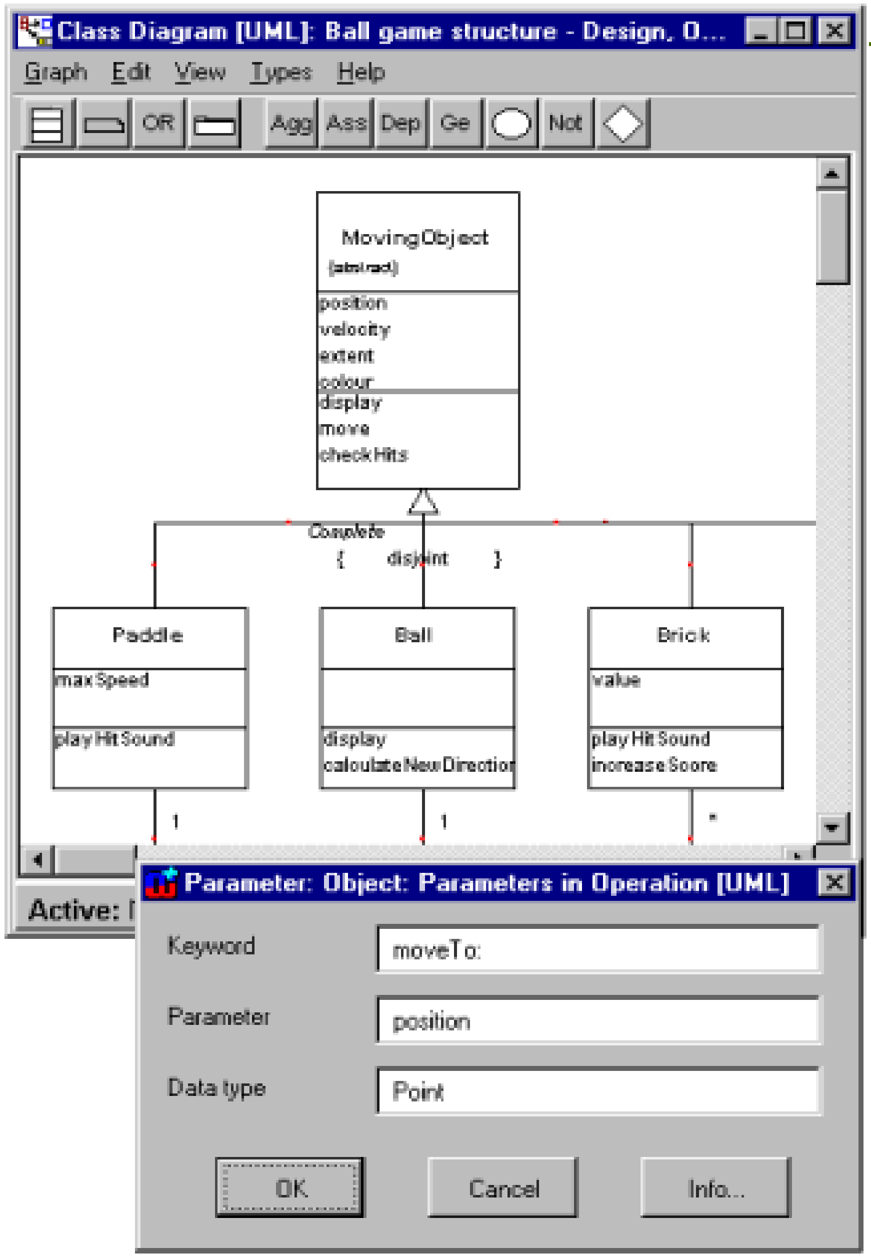
Name	Class [UML]
Ancestor	Object
Project	UML

Local name	Property name	Data type
*Class name	Class name	String
Attributes	Attributes [UML]	Collection: Attribute
Operations	Operations [UML]	Collection: Operatic

Description: An class describes a group of objects with similar properties (attributes), common behaviour



Active: Text Label Condition: Abstract?='T' Grid: 10@10



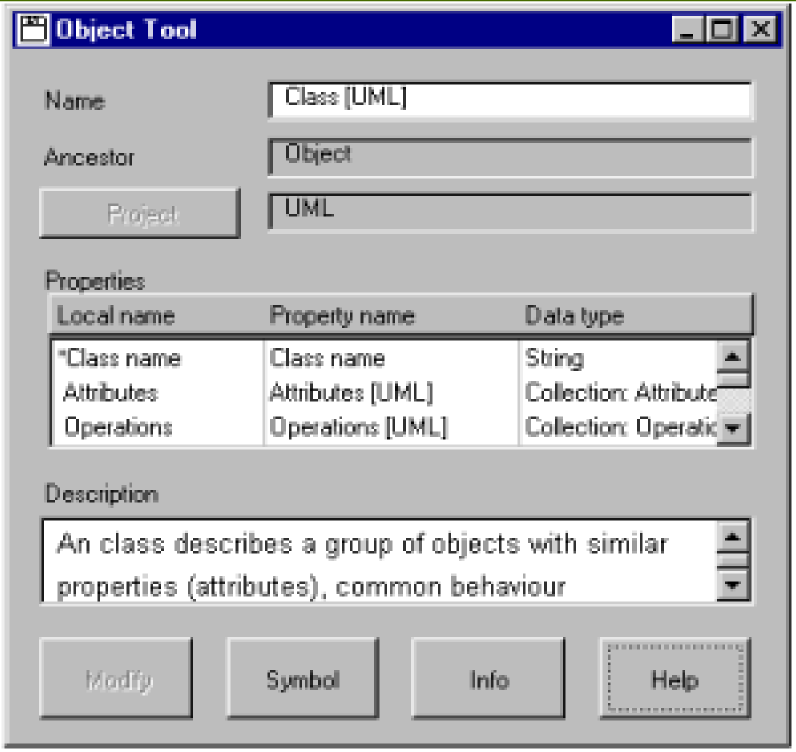
Keyword	moveTo:
Parameter	position
Data type	Point

Quelle: http://www.metacase.com/mwb30index.html

Erstellen eines eigenen CASE-Tools mit MetaEdit+

Entwurf der eigenen Methode

Benutzen der eigenen Methode



Name: Class [UML]

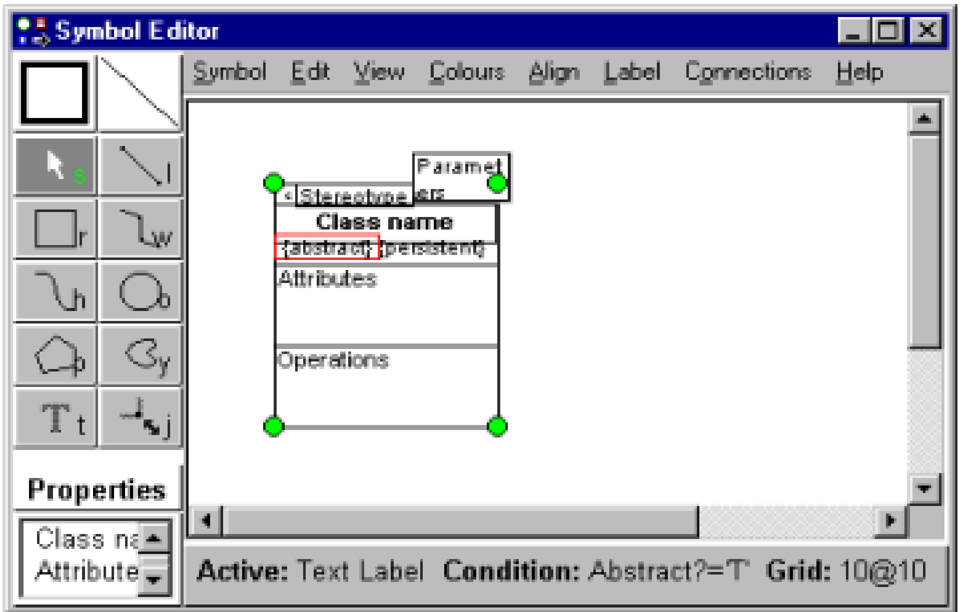
Ancestor: Object

UML

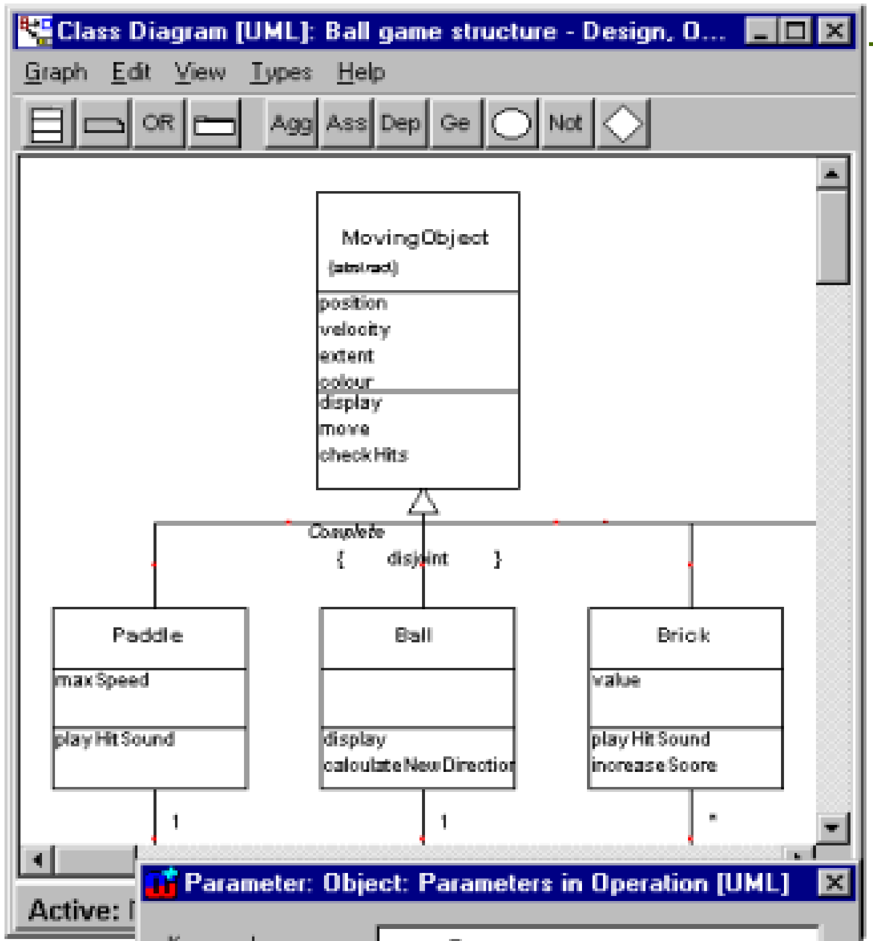
Properties

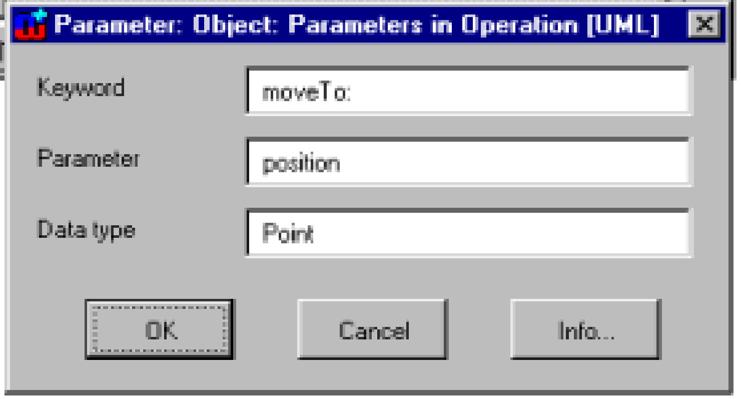
Local name	Property name	Data type
*Class name	Class name	String
Attributes	Attributes [UML]	Collection: Attribute
Operations	Operations [UML]	Collection: Operatic

Description: An class describes a group of objects with similar properties (attributes), common behaviour



Active: Text Label Condition: Abstract?='T' Grid: 10@10





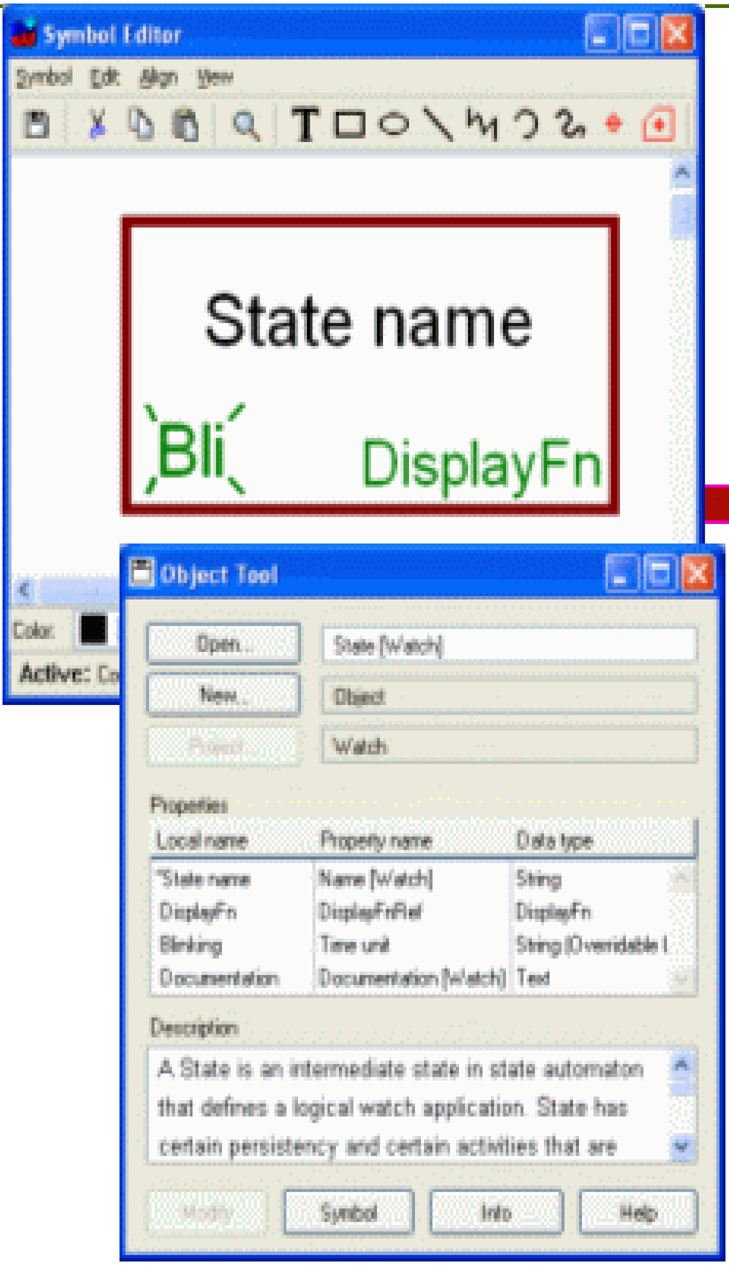
Keyword: moveTo:

Parameter: position

Data type: Point

Quelle: http://www.metacase.com/mwb30index.html

MetaEdit+ Workbench for a State Diagram (STD)



State [Watch]

Object

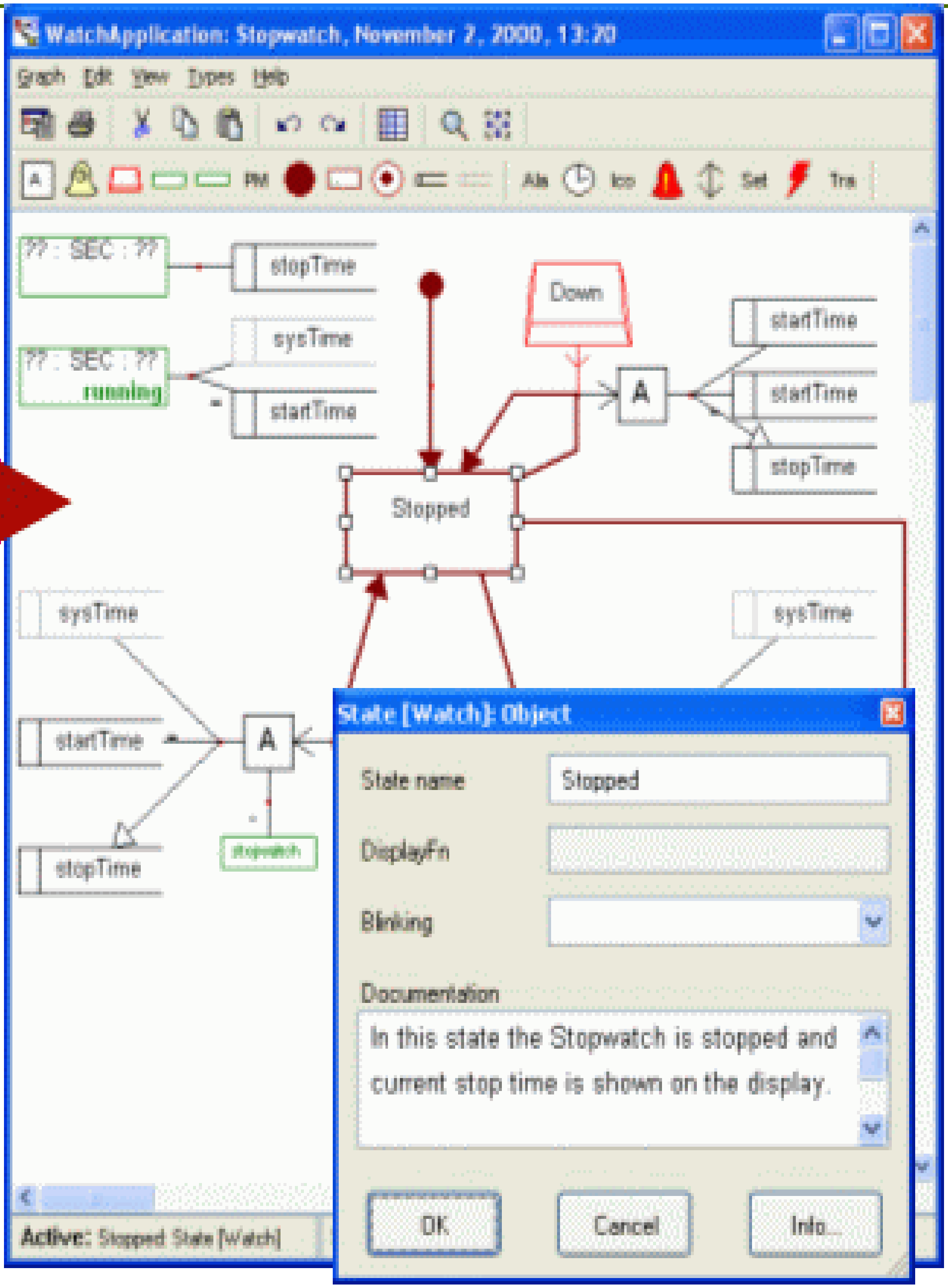
Watch

Properties

Local name	Property name	Data type
State name	Name [Watch]	String
DisplayFn	DisplayFnRef	DisplayFn
Blinking	Time unit	String (Overridable l
Documentation	Documentation [Watch]	Text

Description

A State is an intermediate state in state automaton that defines a logical watch application. State has certain persistency and certain activities that are



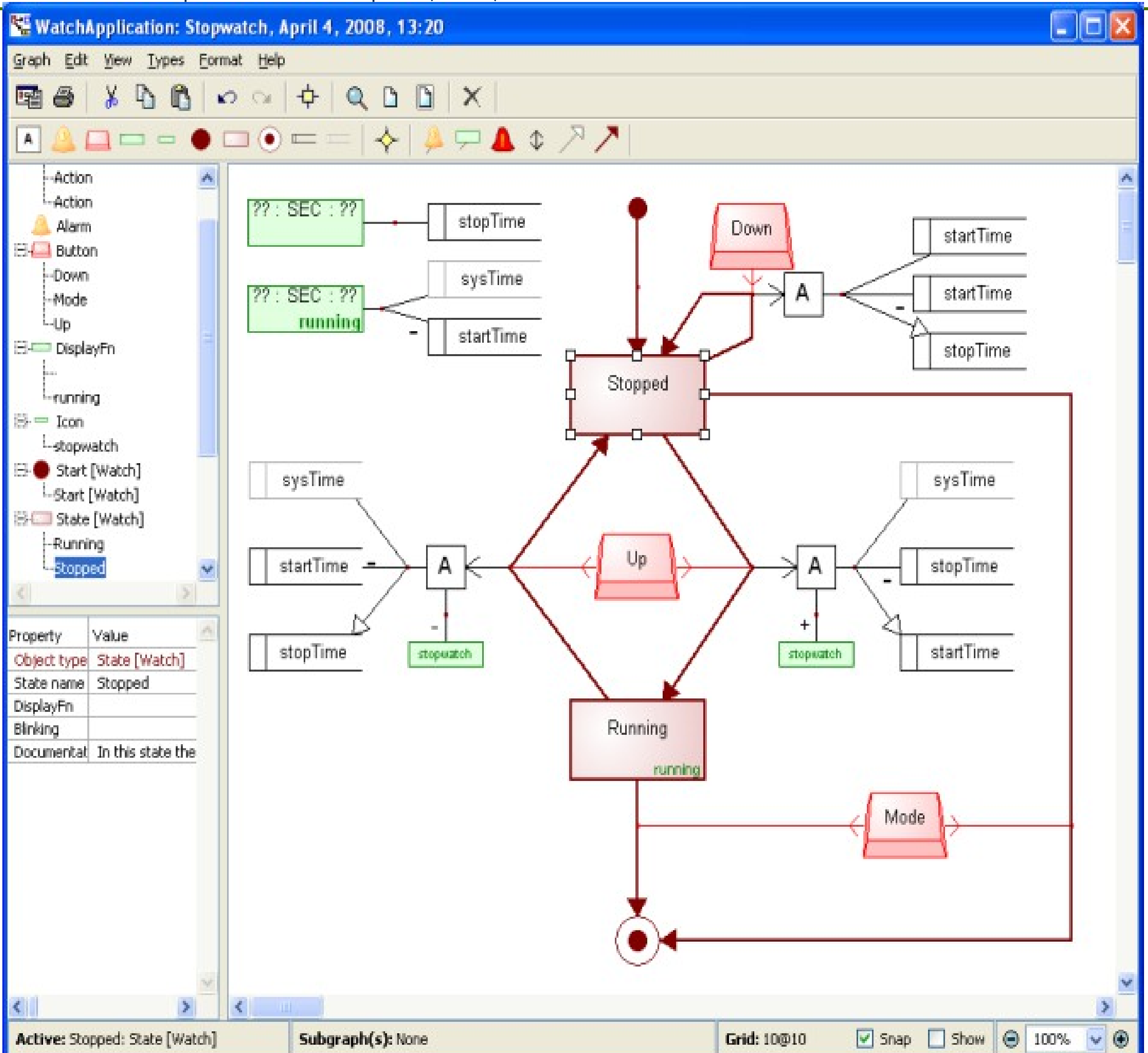
State name: Stopped

DisplayFn:

Blinking:

Documentation: In this state the Stopwatch is stopped and current stop time is shown on the display.

Quelle: http://www.metacase.com/mwb30index.html

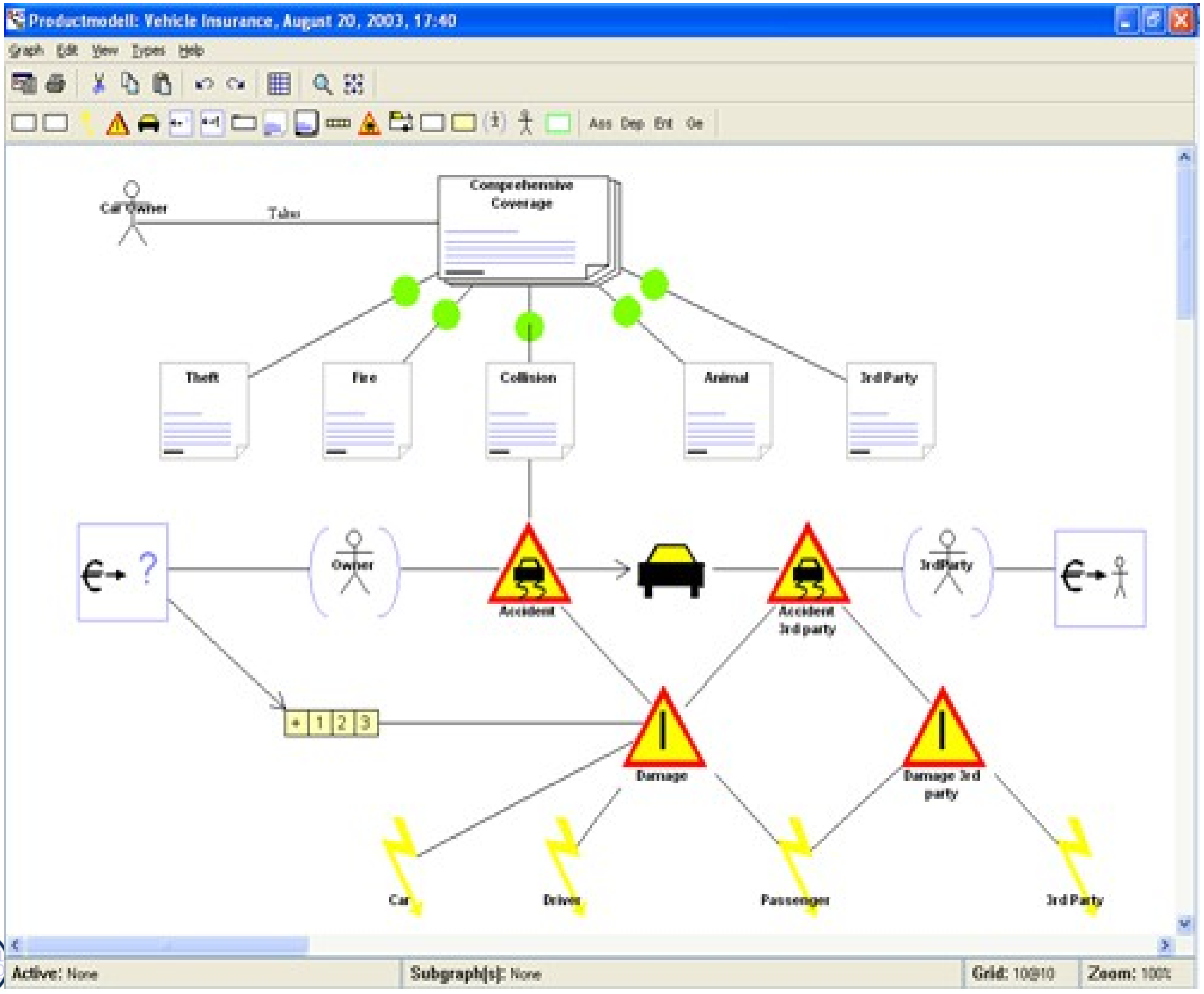


Property	Value
Object type	State [Watch]
State name	Stopped
DisplayFn	
Blinking	
Documentat	In this state the

Active: Stopped: State [Watch] Subgraph(s): None Grid: 10@10 [✓ Snap] Show 100%

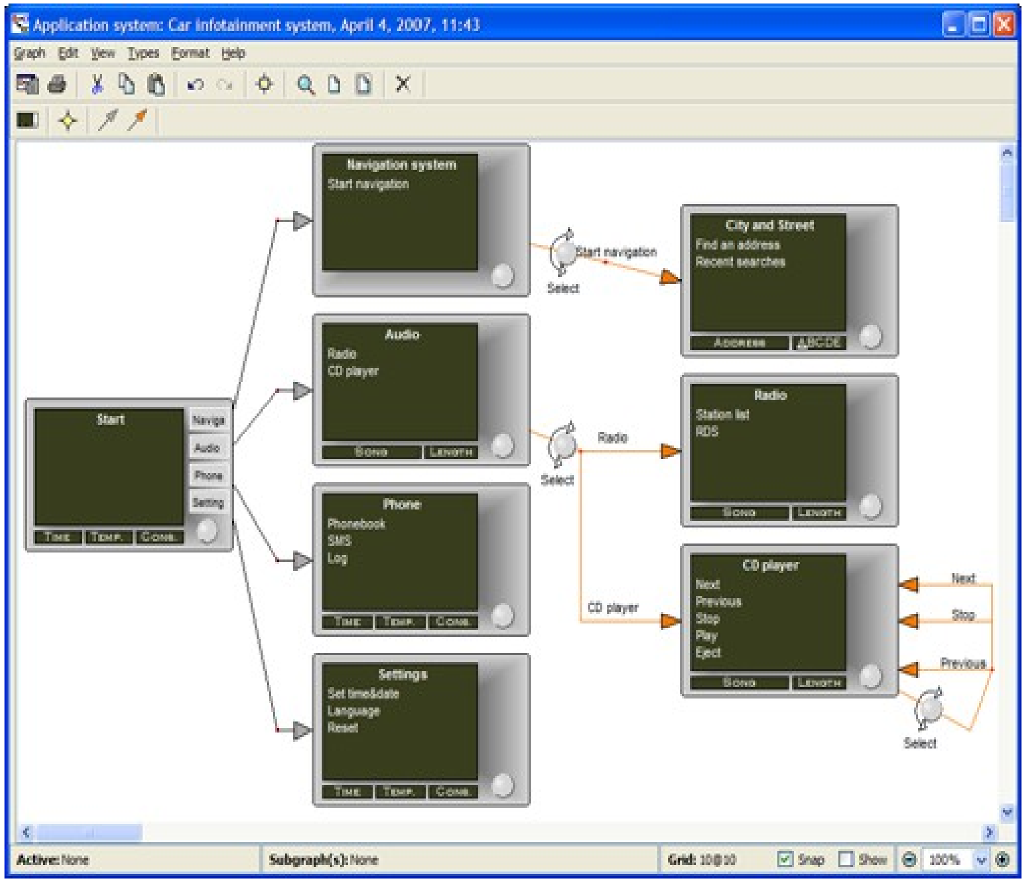
Insurance DSL

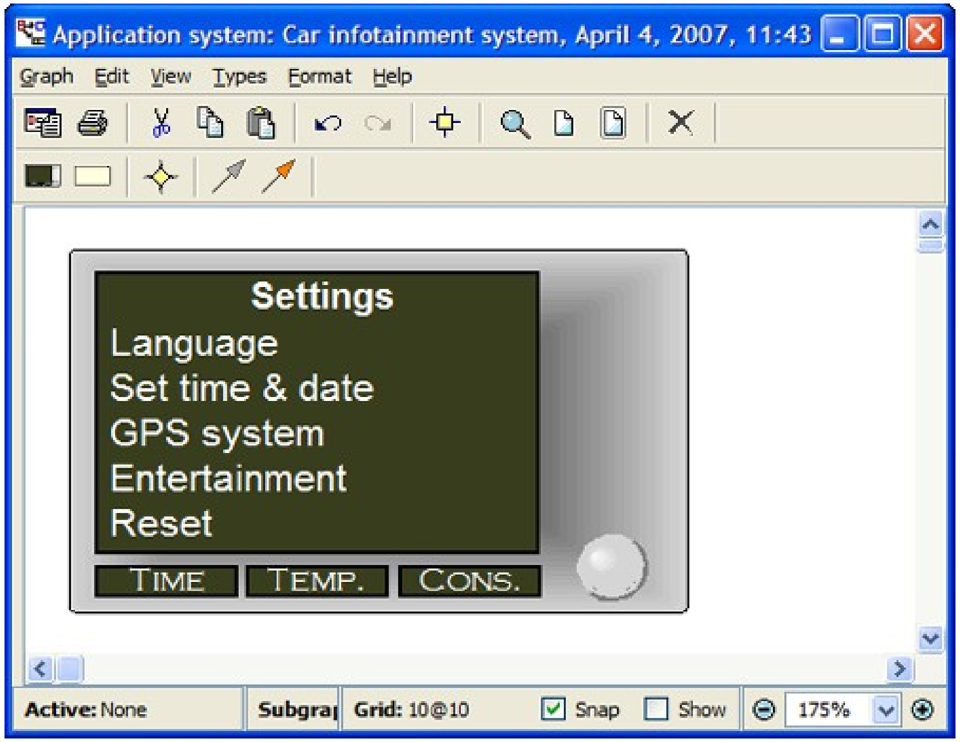
- For modeling of insurance products
- Generators produce the required insurance data and code for a J2EE website



Automotive Entertainment DSL

- Domain: car infotainment system and user interface elements
- Design of the logic and flow via connecting the modeling concepts between GUI and application concept metamodel editor


Application system: Car infotainment system, April 4, 2007, 11:43


Application system: Car infotainment system, April 4, 2007, 11:43

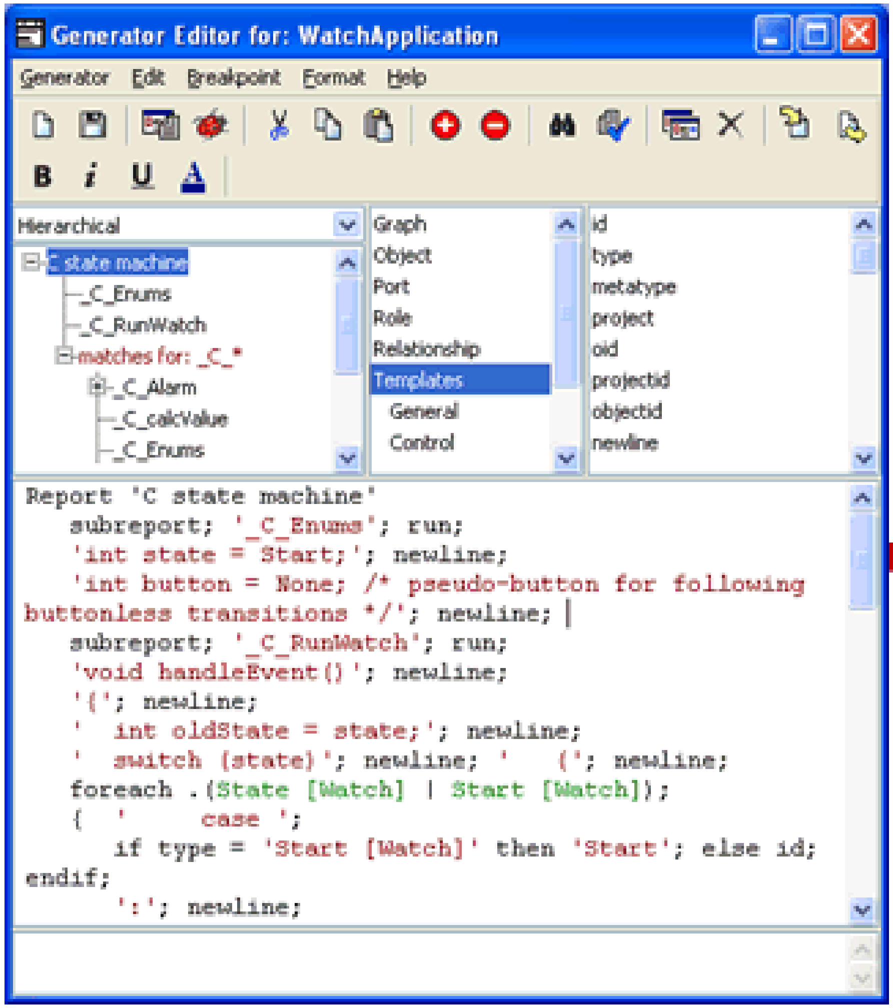
http://www.metacase.com/cases/autoinf

Tools in MetaEdit+

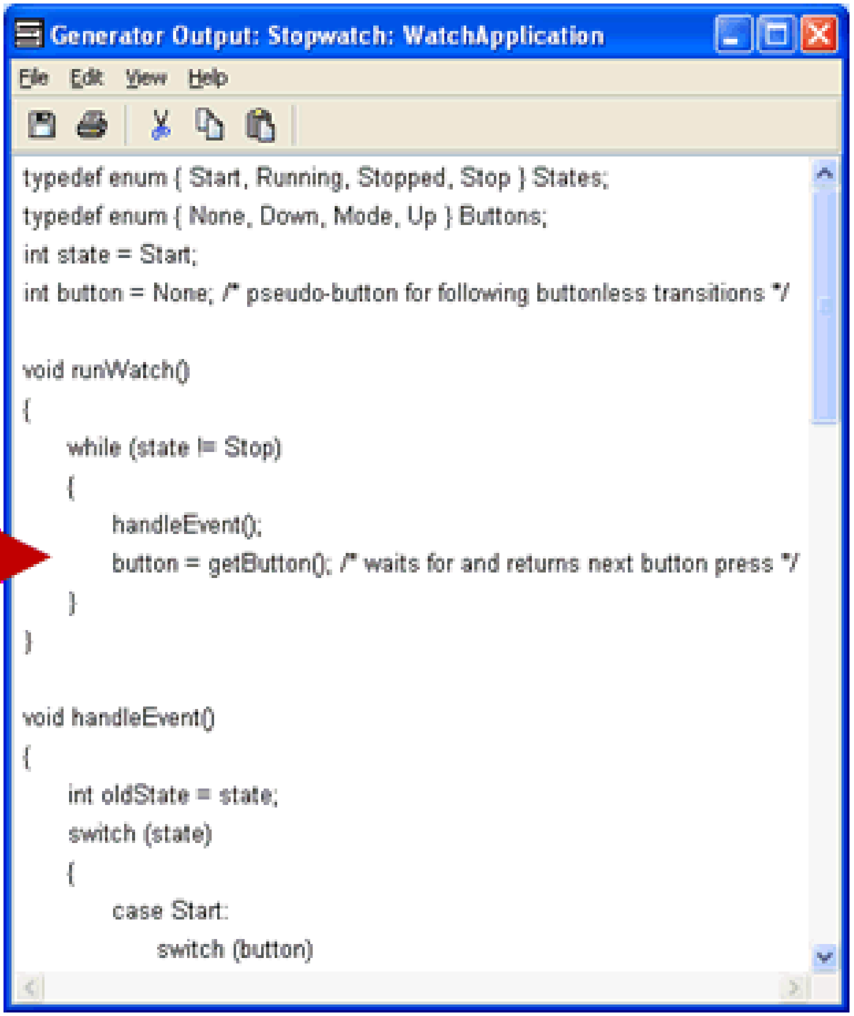
- Report Generator:
  - Script-driven, for the generation of texts and code
- API (API-Server):
  - MetaEdit+ is implemented in Smalltalk
  - Accessible via Web Server (SOAP with WSDL)

```
Report 'ExportToolUIModel'
'<?xml version="1.0" encoding="UTF-8"?>'newline;
'<model>'newline;
foreach .Graph {
  do :Graph {
    if type; = 'Tools UIs Model' then
       subreport; 'ToolUI_XML' run;
    else
       subreport; 'structureXML' run;
    endif
  }
}
'</model>'newline;
endreport
```

Editor for Scripts for Code Generation



```
Report 'C state machine'
    subreport; '_C_Enums'; run;
    'int state = Start;'; newline;
    'int button = None; /* pseudo-button for following
buttonless transitions */'; newline;
    subreport; '_C_RunWatch'; run;
    'void handleEvent()'; newline;
    '{'; newline;
    '  int oldState = state;'; newline;
    '  switch (state)'; newline; '    {'; newline;
    foreach .(State [Watch] | Start [Watch]);
    {  '       case ';
       if type = 'Start [Watch]' then 'Start'; else id;
endif;
       ':'; newline;
```



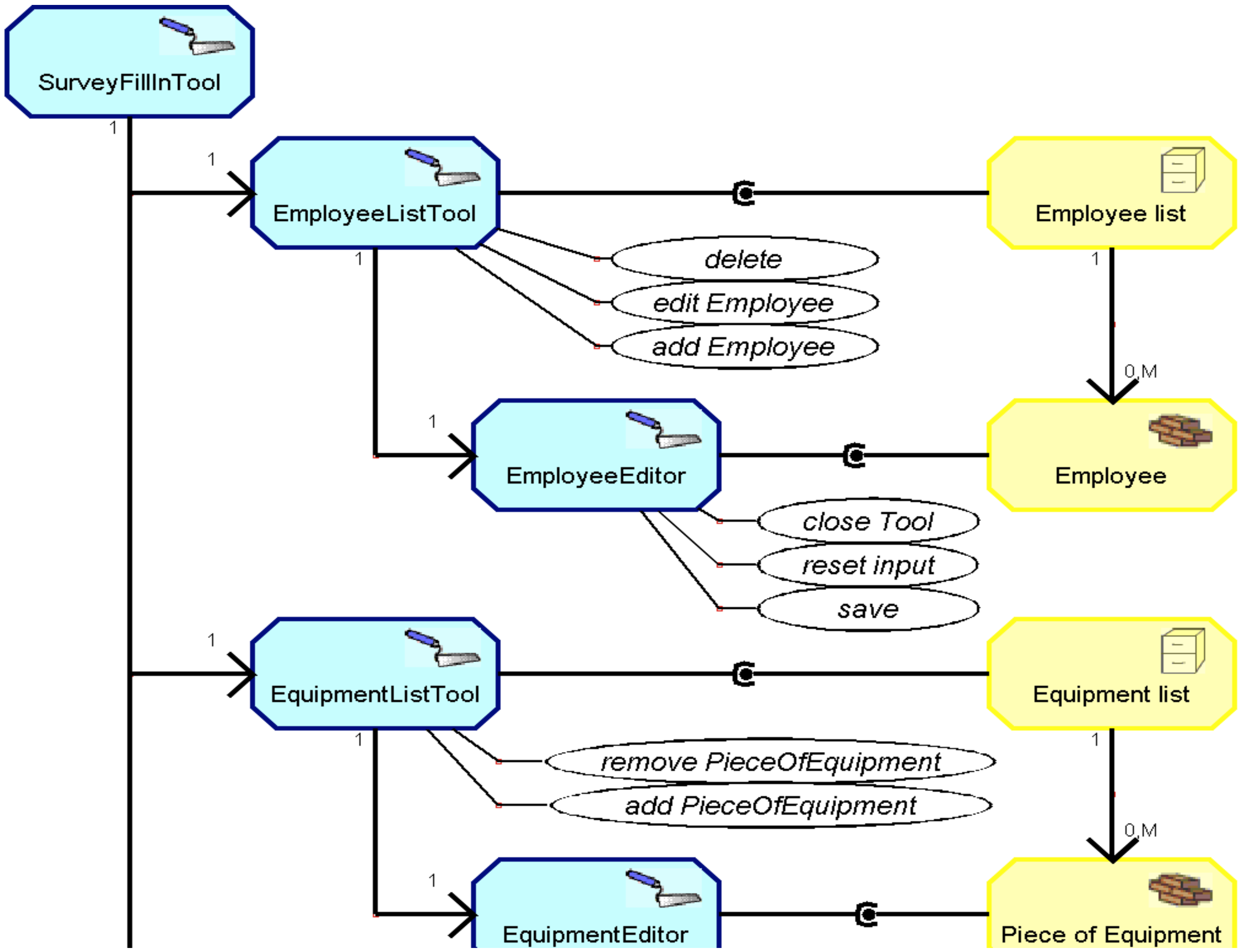
```
typedef enum { Start, Running, Stopped, Stop } States;
typedef enum { None, Down, Mode, Up } Buttons;
int state = Start;
int button = None; /* pseudo-button for following buttonless transitions */

void runWatch()
{
    while (state != Stop)
    {
        handleEvent();
        button = getButton(); /* waits for and returns next button press */
    }
}

void handleEvent()
{
    int oldState = state;
    switch (state)
    {
        case Start:
            switch (button)
```

Tool/Material DSL, Modeled in MetaEdit+

- [Nill] presented a DSL for Tools and Materials (TAM-DSL), modelled in in GOPRR with MetaEdit+
- Editor represents Tools and Materials graphically

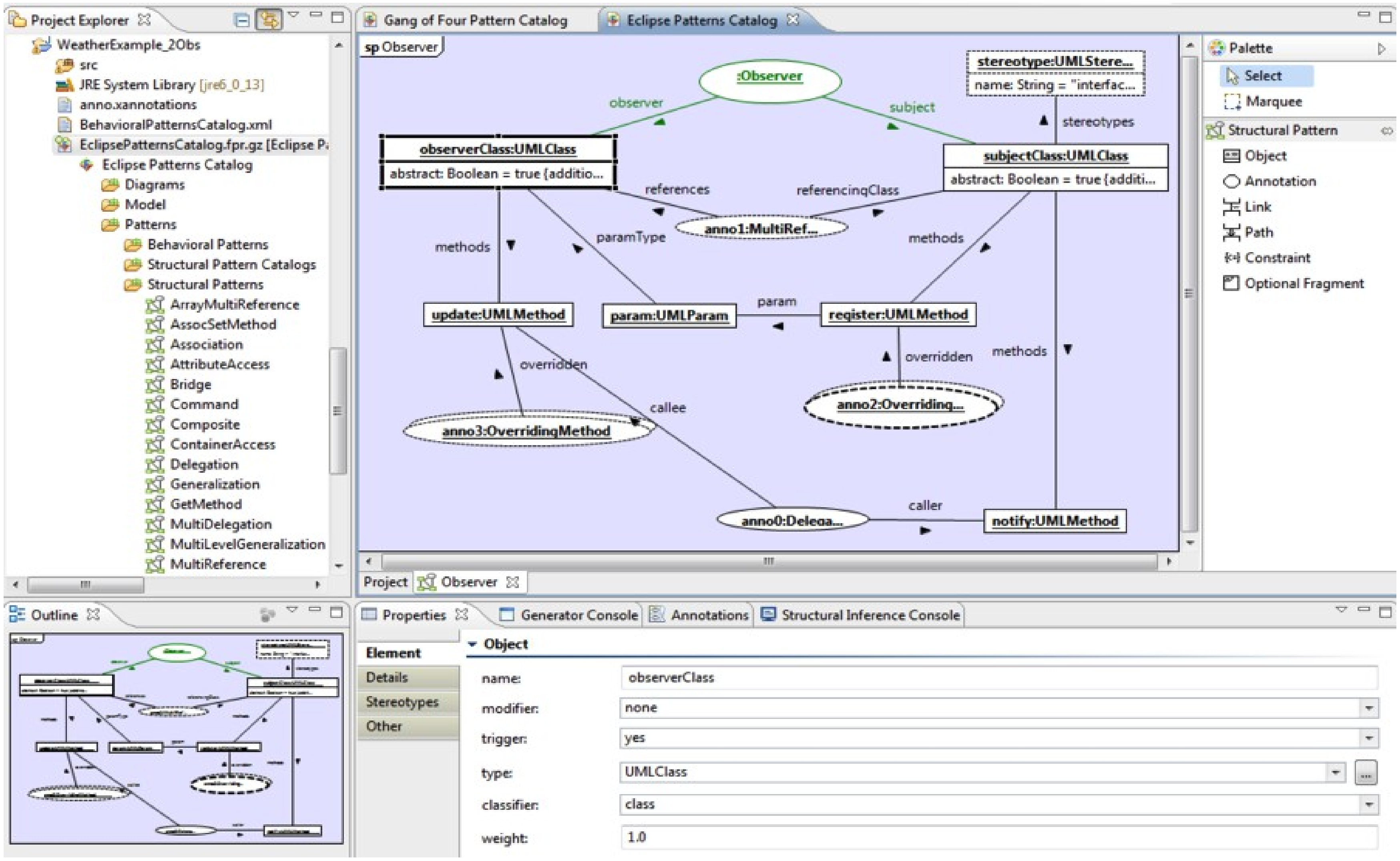


33.3 Introduction to Fujaba

www.fujaba.de

Fujaba

- Fujaba is a MetaCASE-tool based on GRS with home-grown metalanguage and metamodel
- Basic technology: graph pattern matching and rewriting

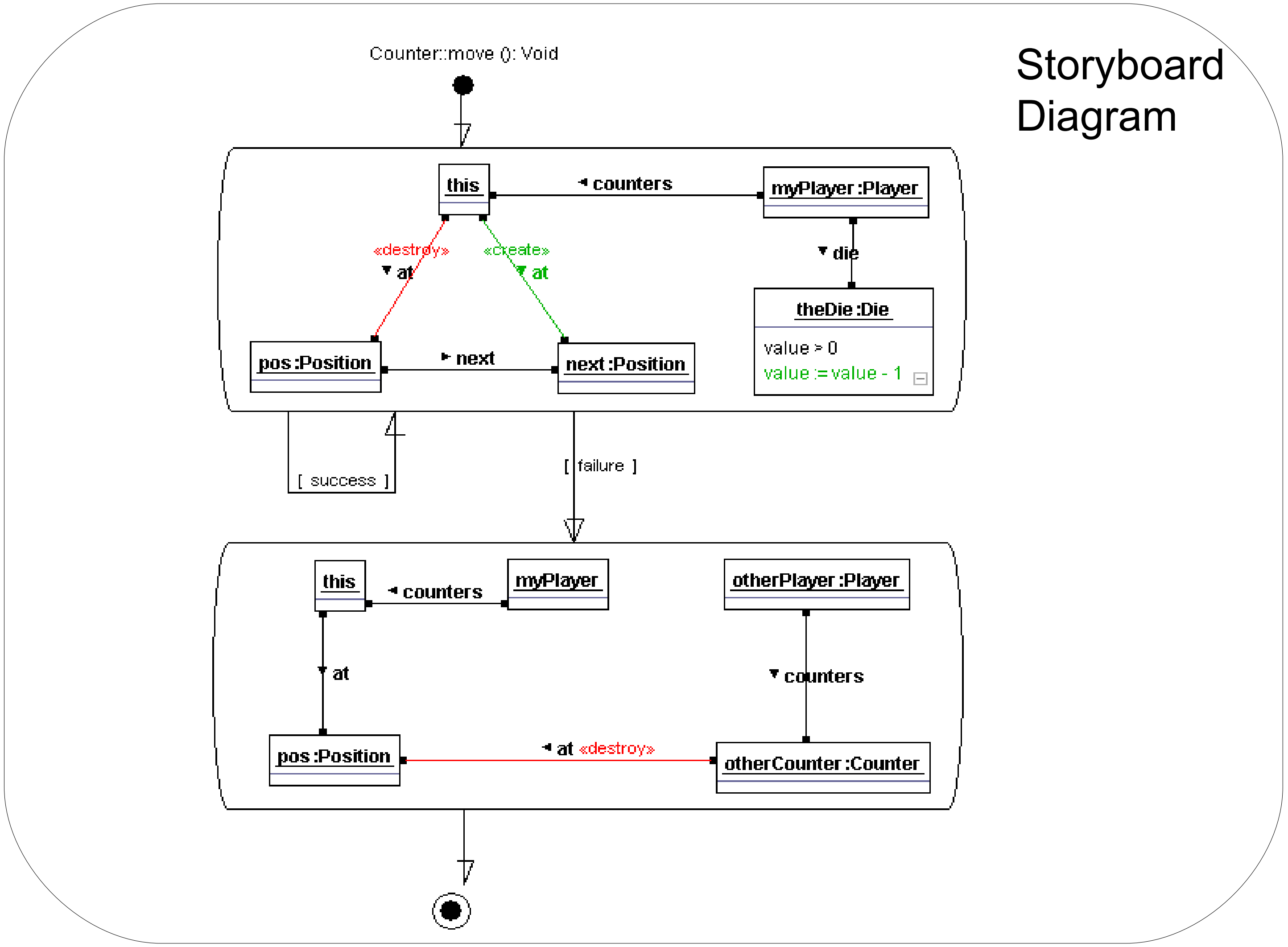


name:	observerClass
modifier:	none
trigger:	yes
type:	UMLClass
classifier:	class
weight:	1.0

http://www.fujaba.de/typo3temp/pics/604c5c6c9e.png

Fujaba Storyboard Diagrams for Adding and Removing Graph Fragments

- Storyboards are activity diagrams in which activities are GRS
- Green color: adding model fragments; Red color: deleting them



Storyboard Diagram

The End