Software Engeneering

Lecture Notes on CASE-Tools: TogetherJ

Vinko Novak (novak@in.tum.de)

Technische Universität München

Institut für Informatik

Friday, 16th Nov. 2001

Outline of the lecture

- What is CASE?
  - **The acronym**
  - **CASE tool**
  - **CASE environment**
  - **Levels of Integration**
  - **Typical functionality and components of CASE tools**

- Working with TogetherJ
  - **Analysis (use cases, sequence diagrams, object models)**
  - **Design (reverse engineering, class models)**
  - **Implementation (forward engineering, roundtrip engineering)**

- Live Demo

What does CASE mean?

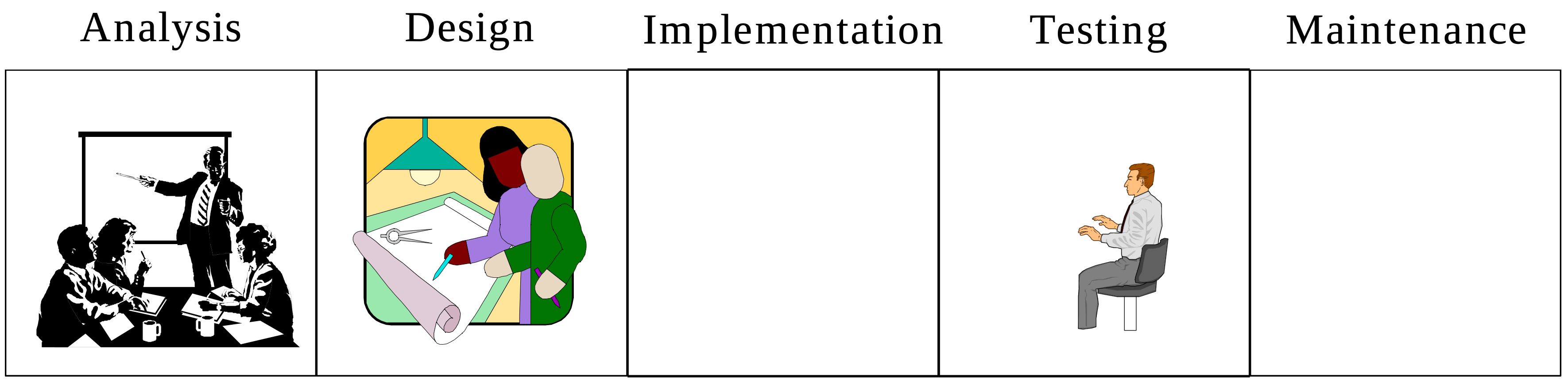
- The acronym CASE stands for
  - **C**omputer
  - **A**ided
  - **S**oftware
  - **E**ngineering
- CASE is the use of computer-based support in the software development process
- different aspects of the Software Development Process (e.g. managerial, administrative, technical etc.)

What is a CASE Tool ?

- A computer-based product aimed at supporting one or more activities within any aspect of the software development process is called **CASE Tool**.

- Tools which support only one particular part of this process (such as compilers, editors, UI generators) are also called CASE tools.

- Usually, CASE tools are defined as browsers and editors for models in graphical and textual form.


Analysis
Design
Implementation
Testing
Maintenance

What is a CASE Environment ?

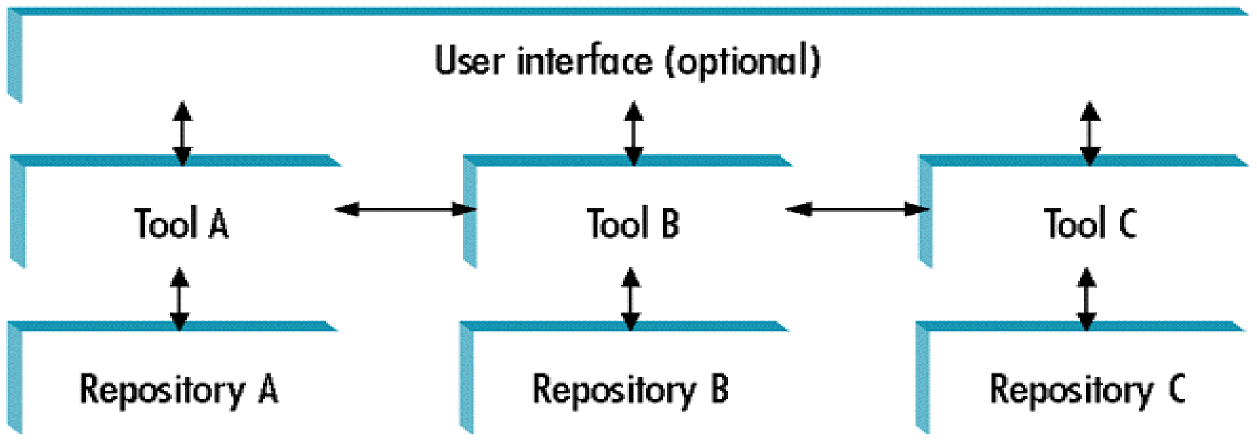
- A CASE *environment* is a collection of CASE tools with an integration approach that supports the interactions that occur among the tools to:
  - **Create models**
  - **Archive models**
  - **Share models**

- The interaction may be done by
  - **common export/import format**
  - **a shared database**
  - **a repository (checkin, checkout)**

Level of integration

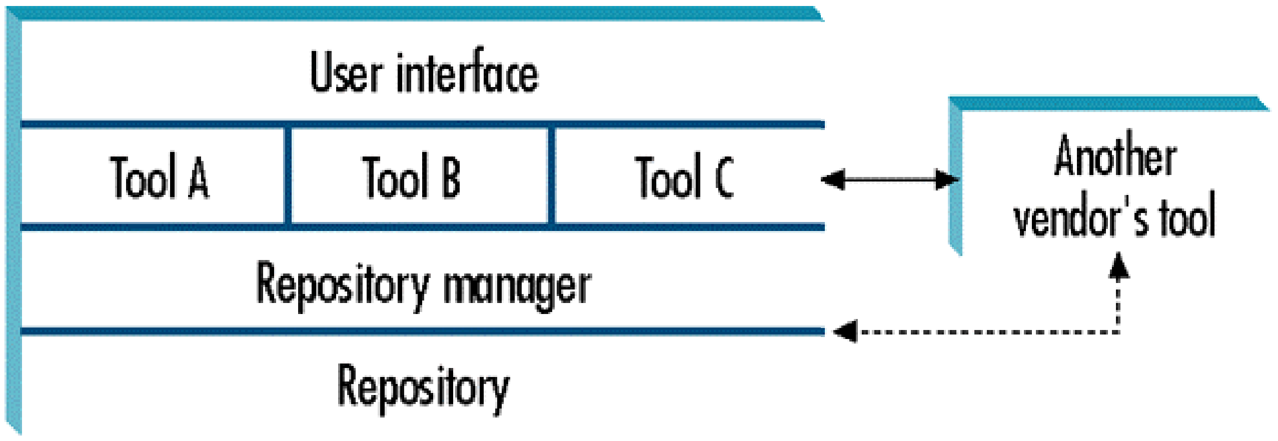
- Distinction between different levels of tool integration
  - **Level 0** ***(tools not integrated)***
  - **Level 1** ***(tools exchange files)***
  - **Level 2** ***(tools provided by single vendor)***
  - **Level 3** ***(tools access shared repository)***

Level of Integration: Level 1 (tools exchange files)



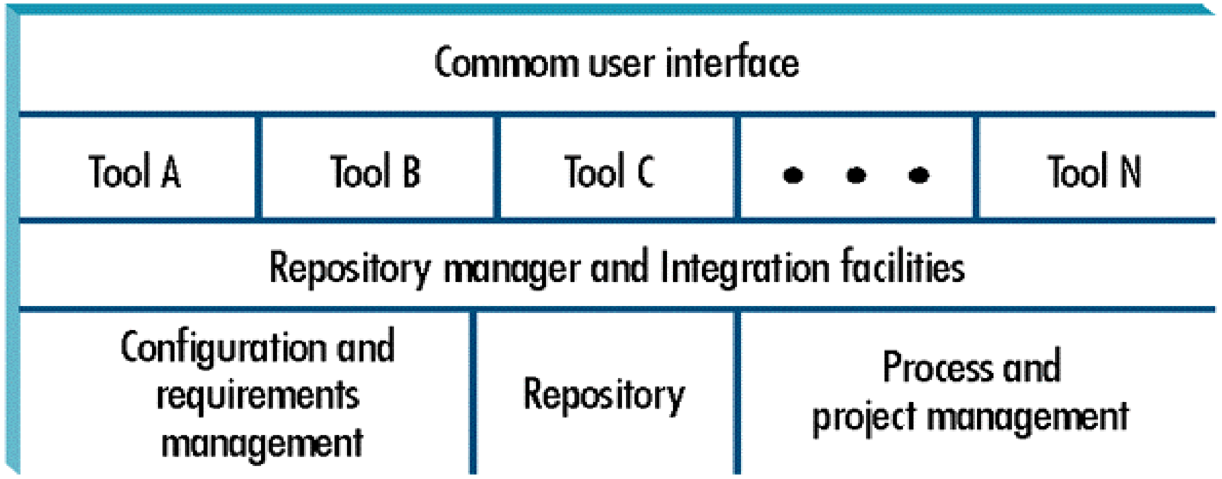
- If the tools were jointly developed, integration can be highly optimized
- Otherwise, tools exchange models only through import and export

Level of Integration: Level 2 (tools provided by single vendor)



- the environment is tightly integrated and optimized
- however, attempts to integrate a tool offered by another vendor results in the same issues as in level 1

Level of Integration: Level 3 (tools access shared-repository)


Commom user interface
Tool A
Tool B
Tool C
• • •
Tool N
Repository manager and Integration facilities
Configuration and requirements management
Repository
Process and project management

- any tool from any vendor can access any model with common services

Functionality of CASE tools

- Core functionality
  - **Browsing and editing of models with a graphical user interface**
  - **Automatic code generation**
  - **Automatic documentation generation**
- Additional functionality
  - **Consistency checks between diagrams and the underlying models**
  - **Support the whole software life cycle**

Typical components of CASE tools

- Project repository
  - **persistent storage of all development documents**
    - **Mockups, RAD, SDD, ODD, Meeting Protocols, Source Code**
  - **integrated version control system**
  - **concurrent, distributed modeling**
- Interface to other tools
  - **software development tools**
  - **scripting language**

Outline of the lecture

- What is CASE?
  - **The acronym**
  - **CASE tool**
  - **CASE environment**
  - **Typical functionality and components of CASE tools**

- Working with TogetherJ
  - **Analysis (use case diagrams, sequence diagrams, object models)**
  - **Design (reverse engineering, class models)**
  - **Implementation (forward engineering, roundtrip engineering)**

- Live Demo

Together

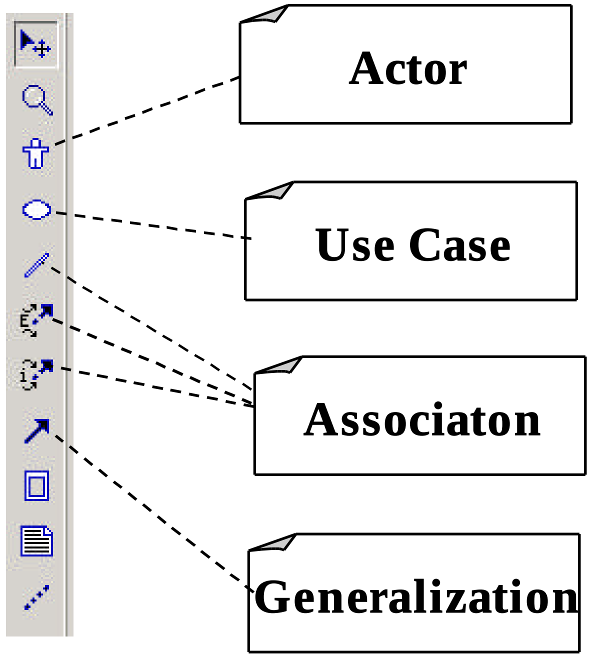
- supports UML 1.3
- supports Java, C++
- supports CVS integration
- supports forward and reverse engineering
- supports generation of documentation from the models
- provides online help and sample applications
- for more documentation visit http://www.togethersoft.com/services/practical_guides/
- written in Java (Windows, Linux, Mac, ...)

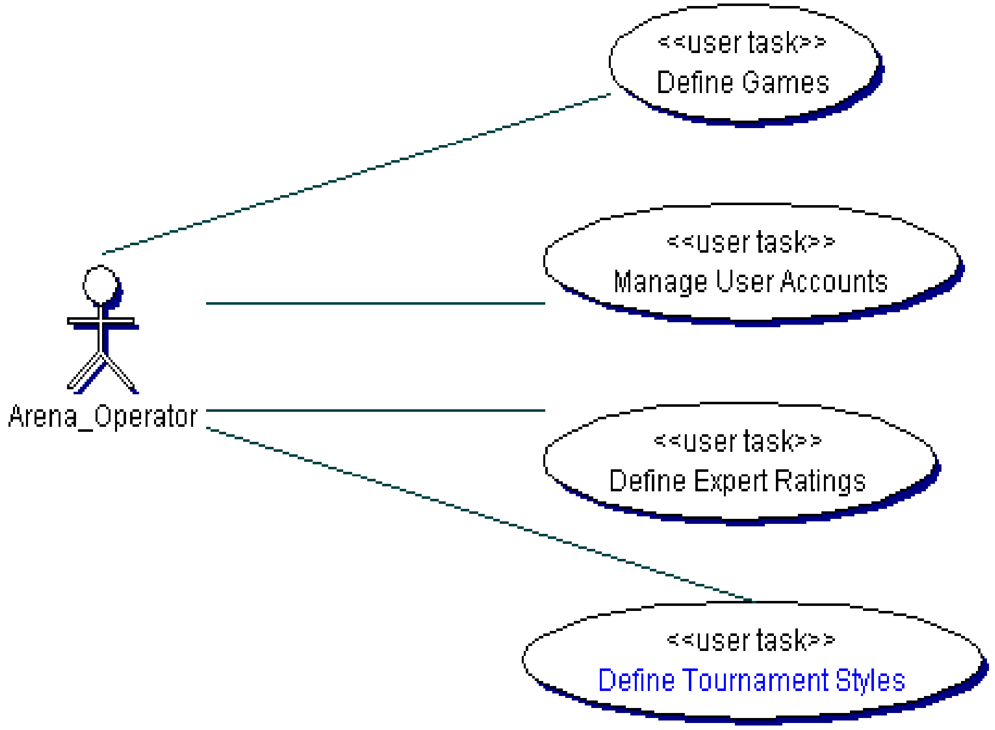
- A free version (whiteboard edition) is available at

  `www.togethersoft.com`

Analysis (use case diagrams)

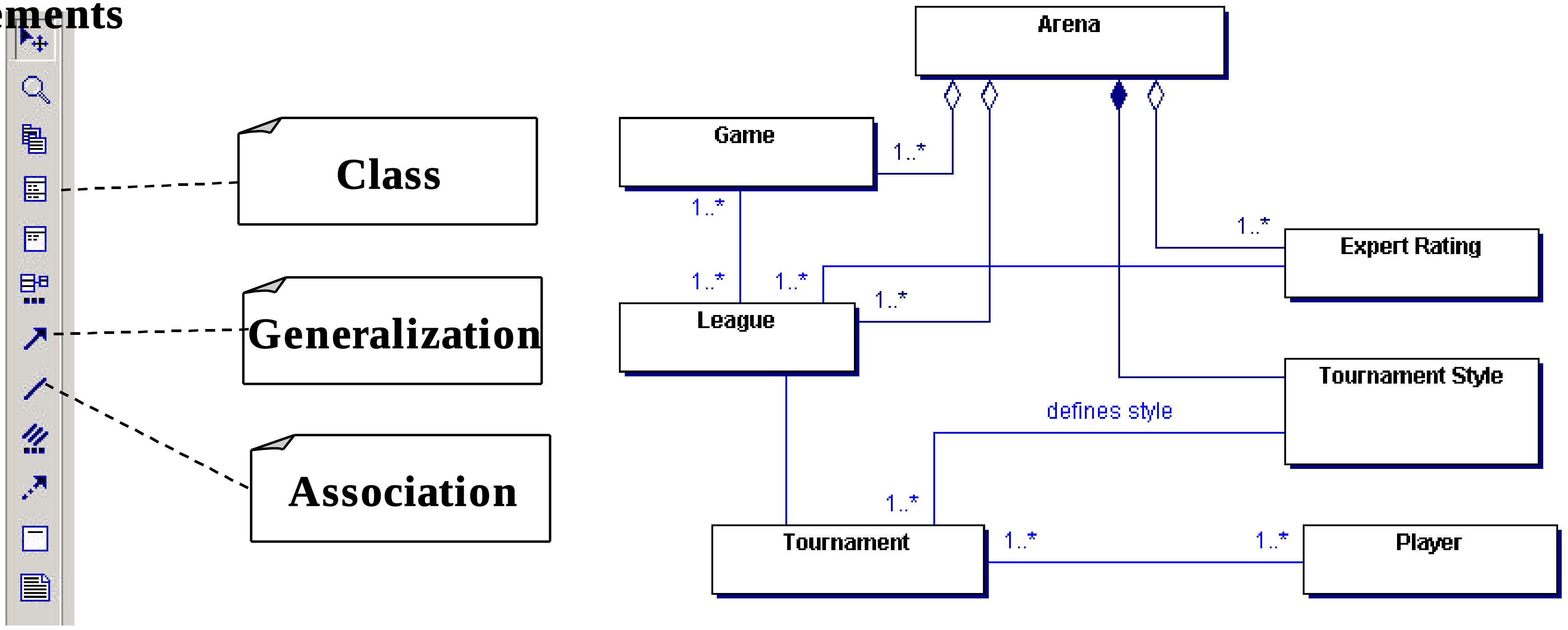
- Use ToolBar to create diagram elements





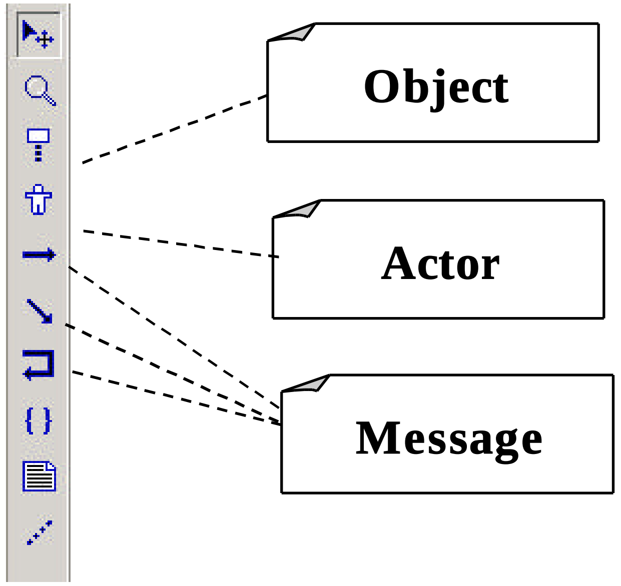
Analysis (class diagrams)

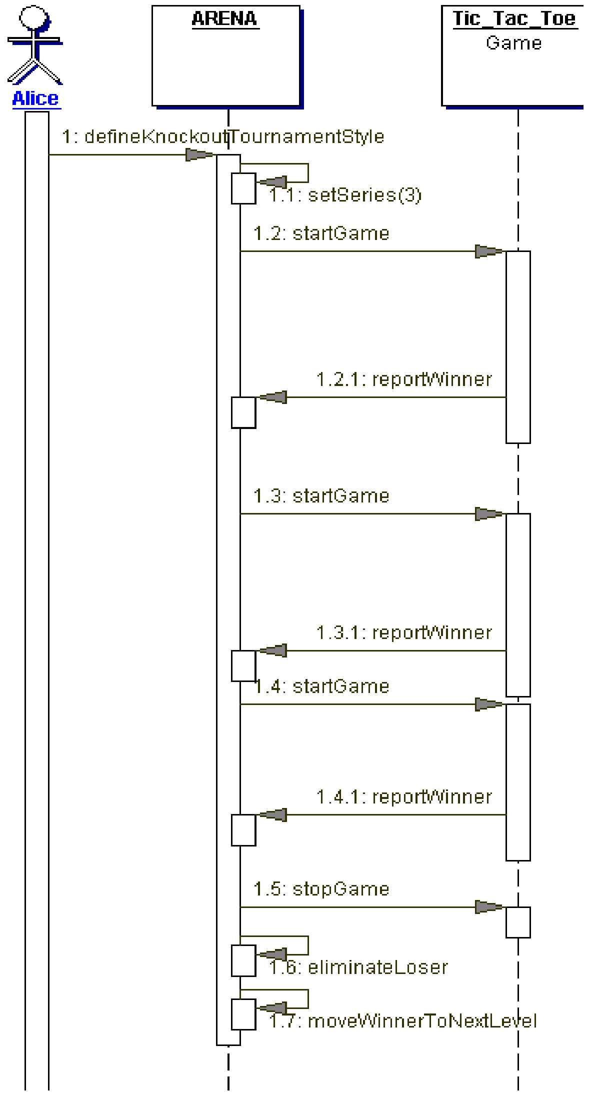
- Use ToolBar to create diagram elements



Analysis (sequence diagrams)

- **Use ToolBar to create diagram elements**



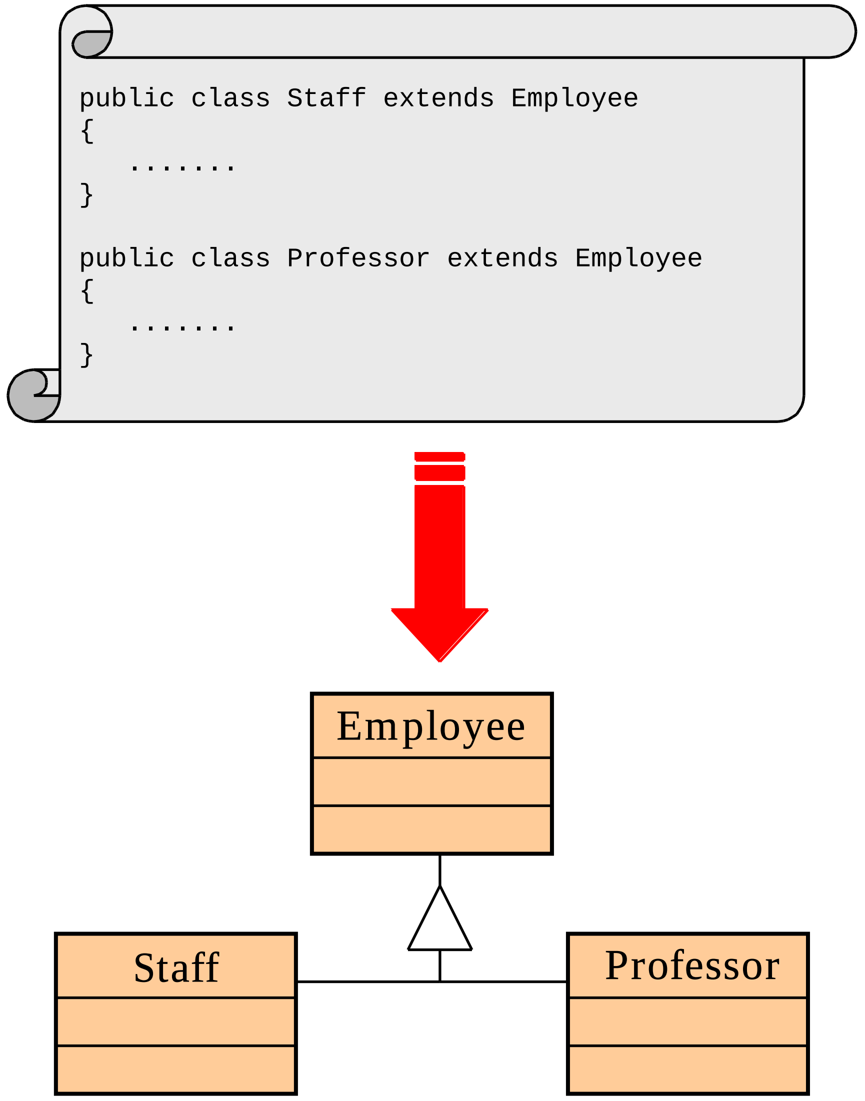


Reverse Engineering

- Reverse engineering is the recreation of an analysis or design model from existing code.

- Typical flow of events

  - **Scan a set of already existing source code files**
  - **Generate the object model for these files**
  - **Allow now modifications on this object model**

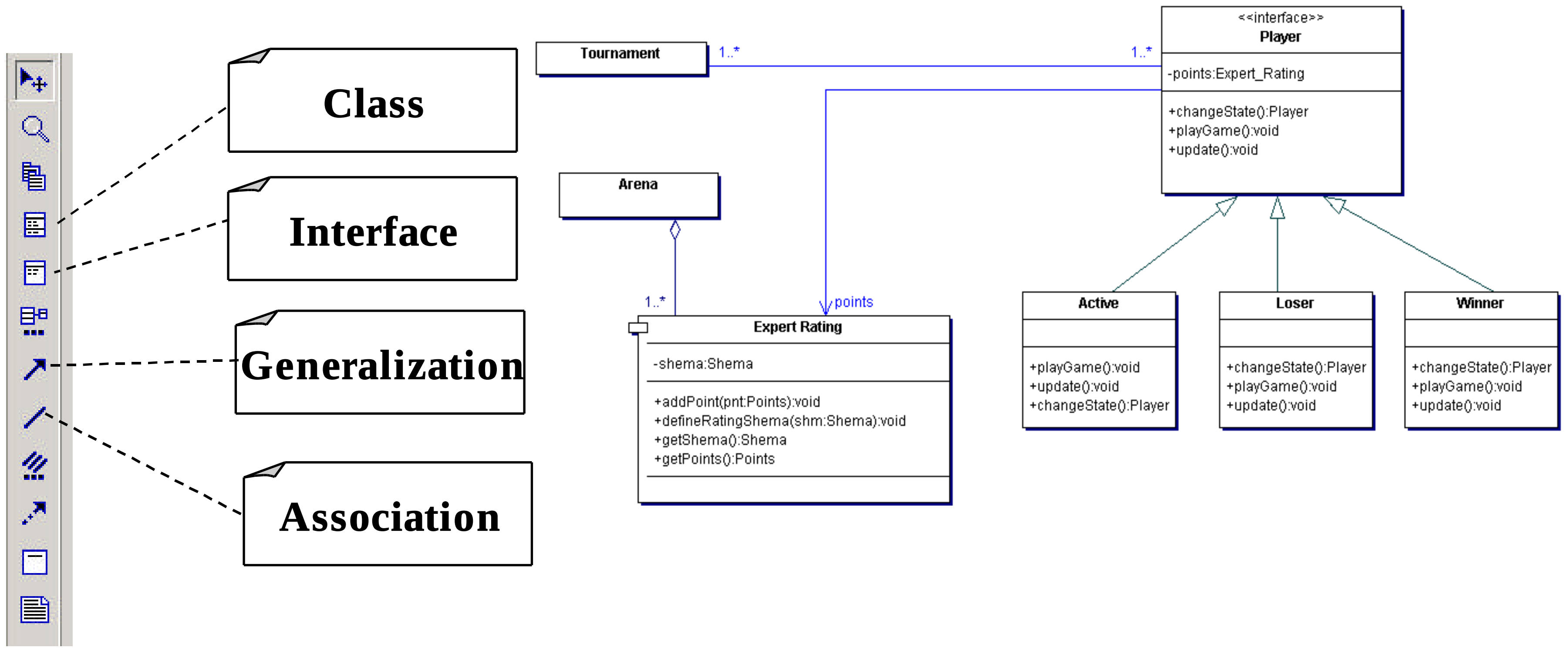


```
public class Staff extends Employee
{
   .......
}

public class Professor extends Employee
{
   .......
}
```

Object Design (class models)

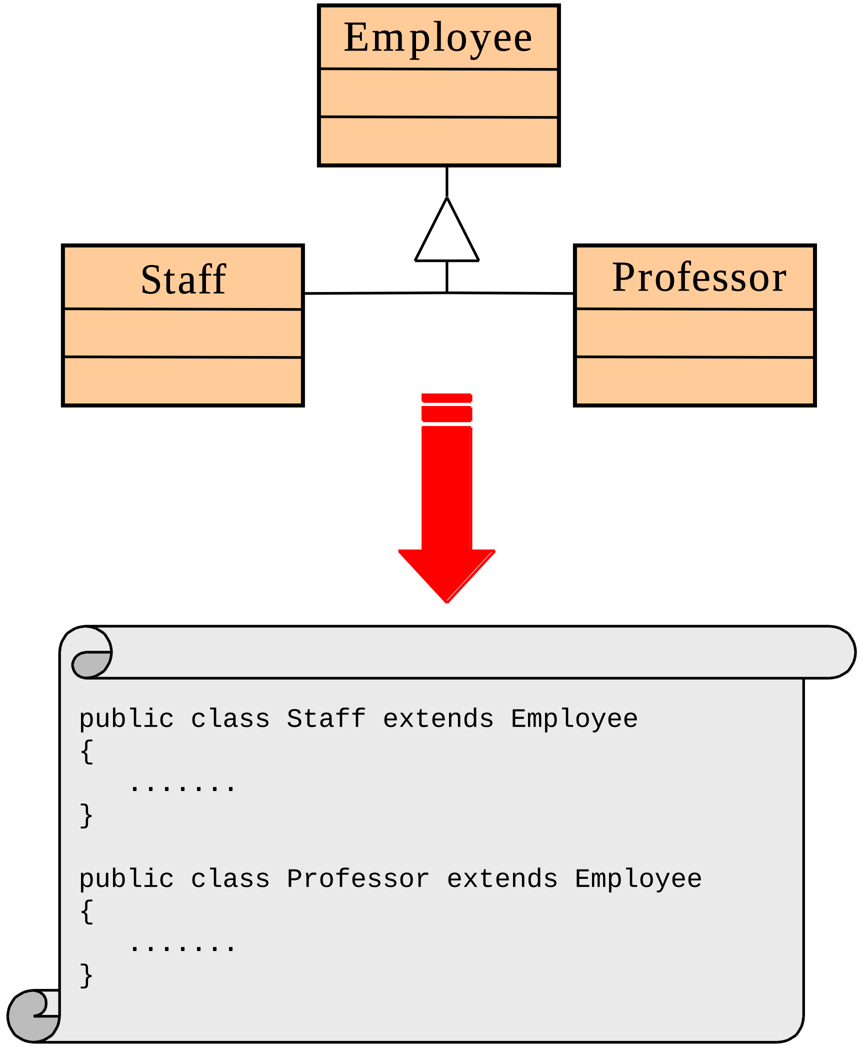
- **Use ToolBar to create diagram elements**



Implementation (Forward Engineering)

- Forward engineering is the generation of skeleton code out of the analysis or design models.
  The developer still has to write the bodies of the methods.

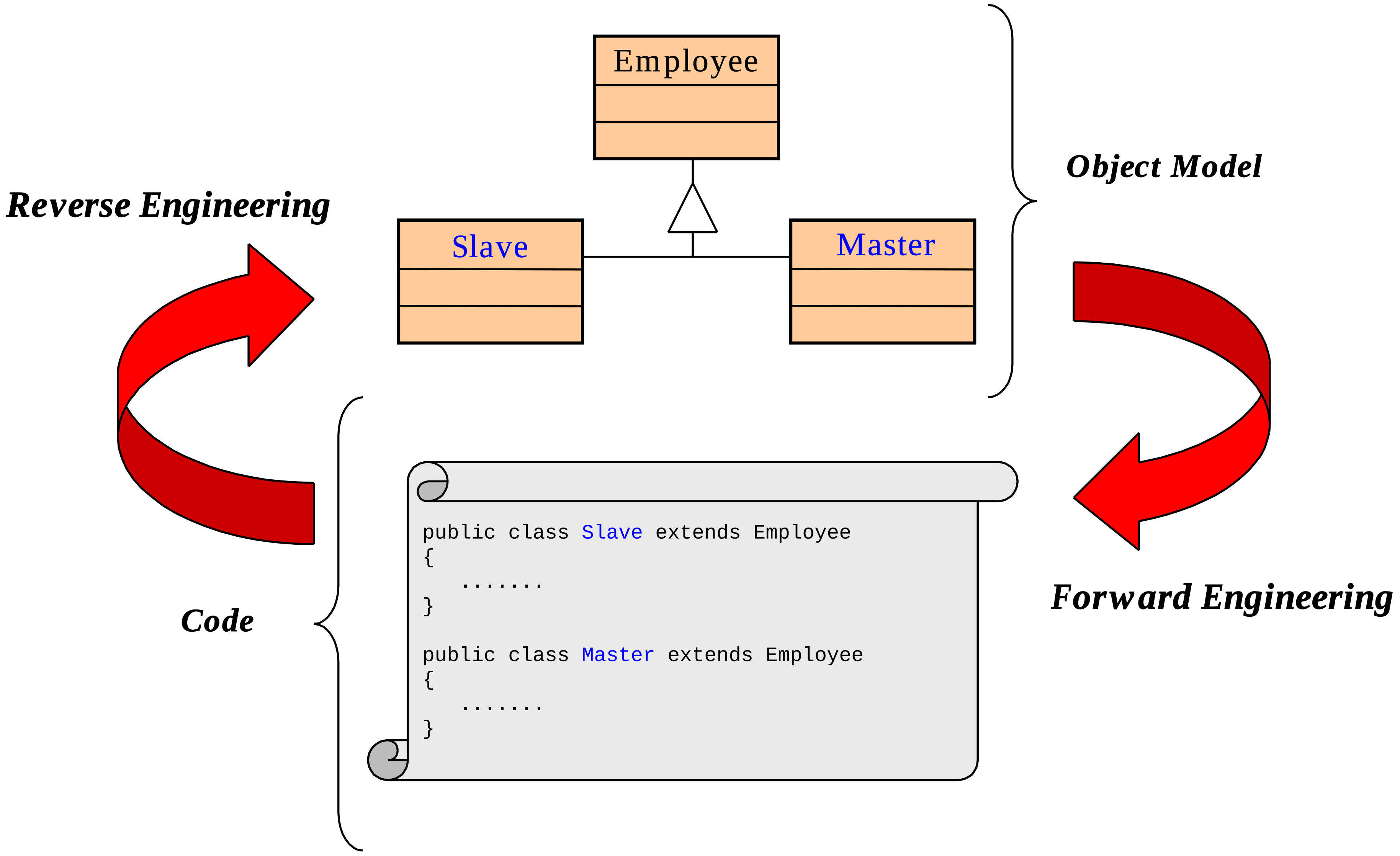
- Typical flow of events
  - **Create or modify an object model for a system**
  - **Generate the code for this model**
  - **Allow external modification of this code**



```
public class Staff extends Employee
{
   .......
}

public class Professor extends Employee
{
   .......
}
```

Implementation (Roundtrip Engineering)



```
public class Slave extends Employee
{
   .......
}

public class Master extends Employee
{
   .......
}
```

Online Demo