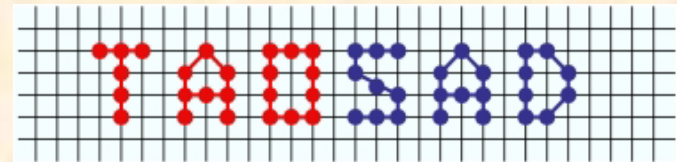


# Introduction to Early Aspects



**Bedir Tekinerdogan**

**Bilkent University,  
Department of Computer Engineering  
Bilkent, Turkey, Ankara  
e:mail - [bedir@cs.bilkent.edu.tr](mailto:bedir@cs.bilkent.edu.tr)  
<http://www.cs.bilkent.edu.tr/~bedir/>**



## About this presentation

- This presentation is a reproduction of our earlier presentation at the 2<sup>nd</sup> AOP Workshop in the ECOOP 1998 Conference:
- **B. Tekinerdogan & M. Aksit. Deriving design aspects from conceptual models. In: Demeyer, S., & Bosch, J. (eds.), Object-Oriented Technology, ECOOP '98 Workshop Reader, LNCS 1543, Springer-Verlag, pp. 410-414, 1999.**

# Current aspect identification approaches

- In general ad-hoc
  - “aspects are for the picking...”
- Usually based on code-analysis:

“Assume the given example....

Concerns crosscuts code and is tangled in classes...”

```
class Point {
    void setX(int x) {
        DisplayTracker.updatePoint(this);
        Tracer.traceEntry("Point.set");
        _x = x;
        Tracer.traceExit("Point.set");
    }
}
```

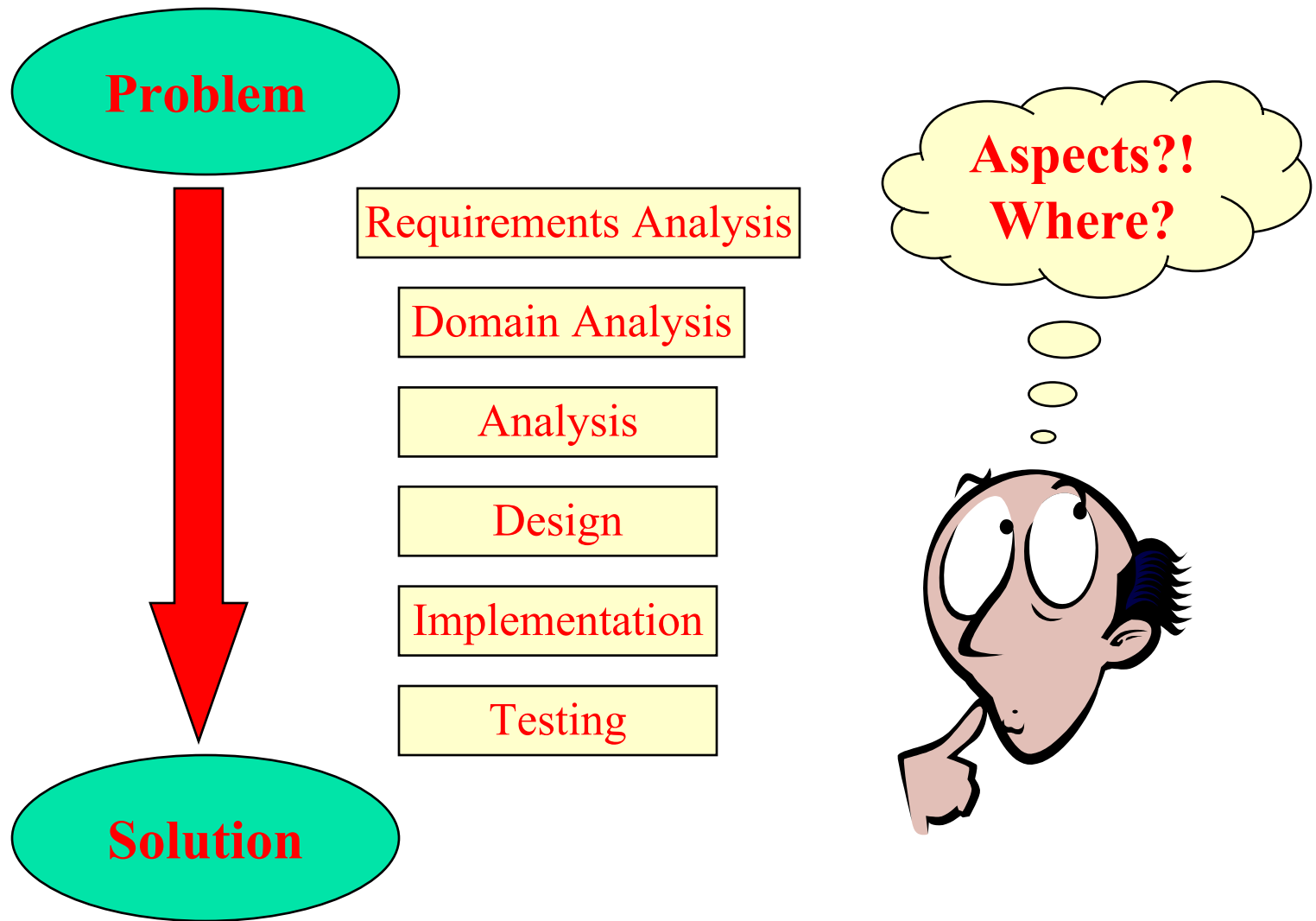
## Characteristics of an aspect...

 **Systemic, tends to affect multiple components**

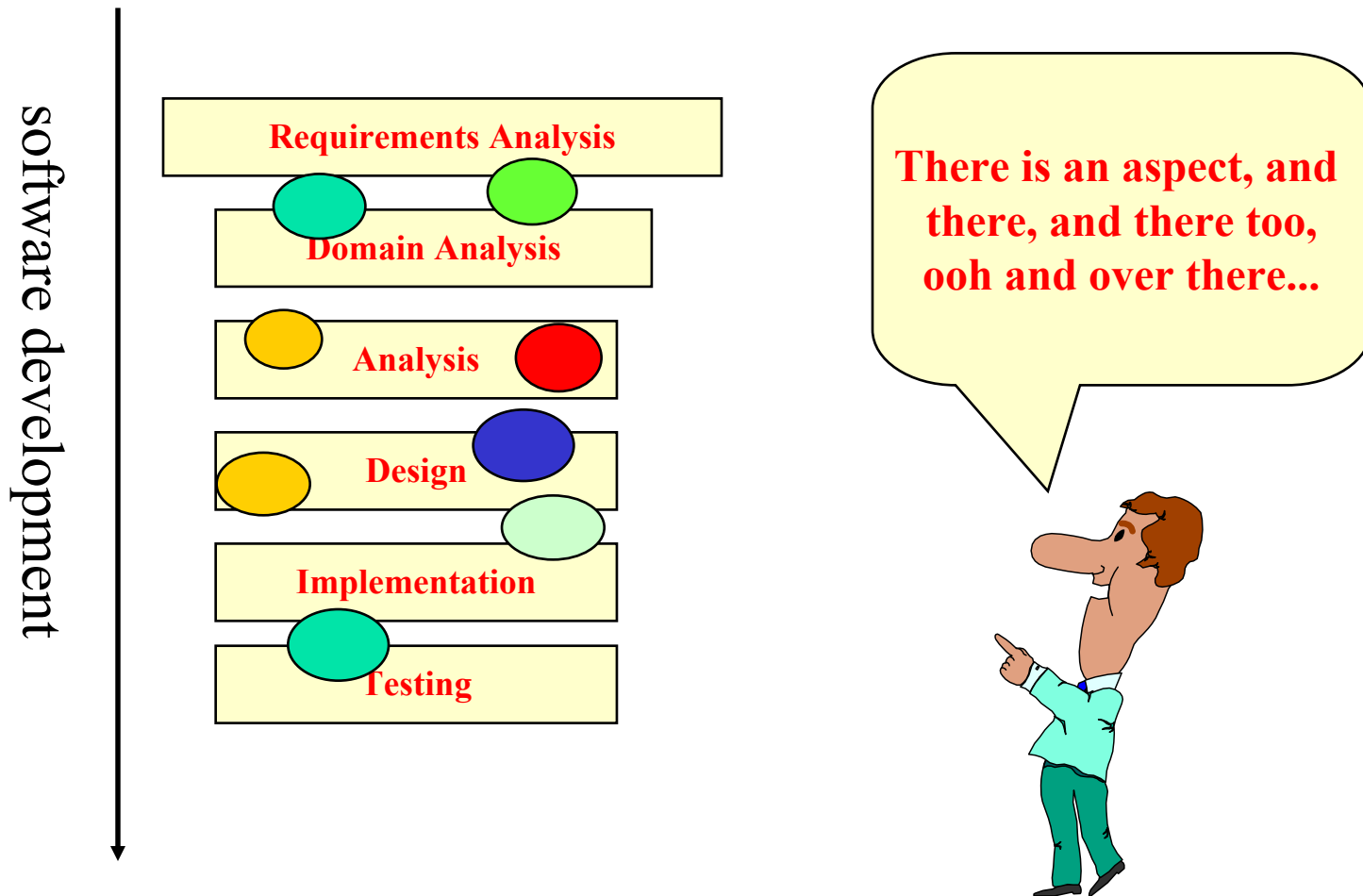
 **Cross-cuts, spread over different components**

 **Affects performance/quality**

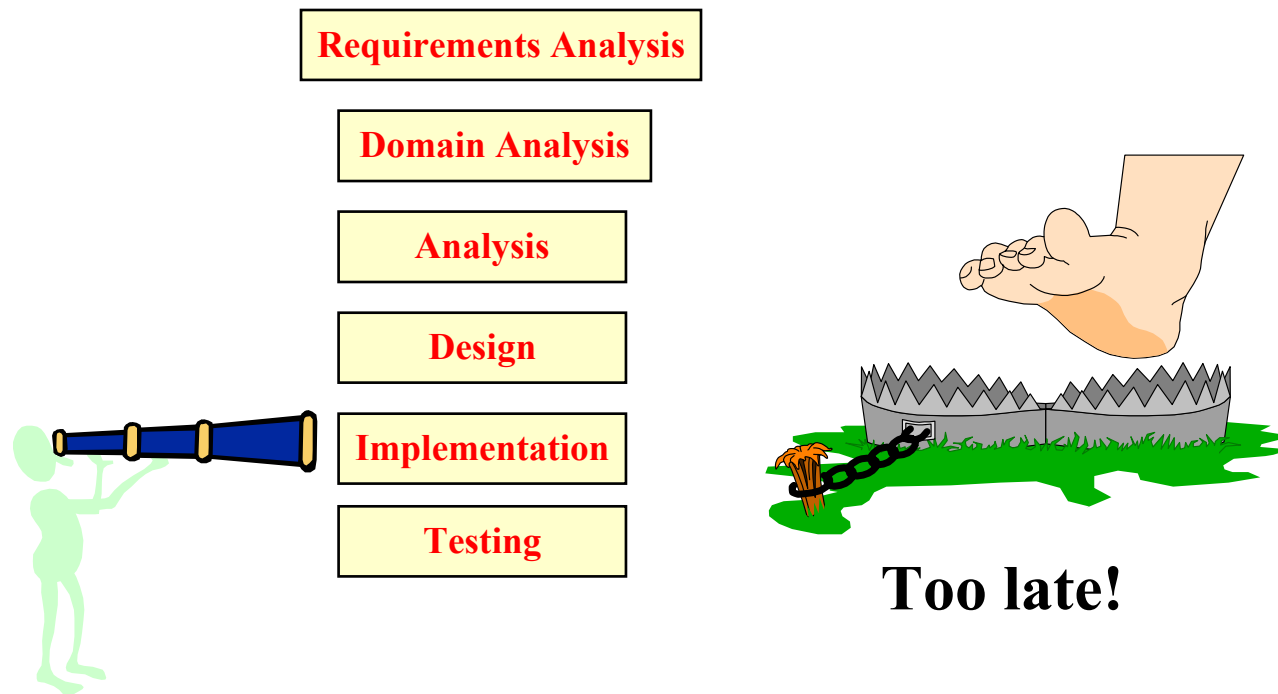
# Aspect identification...



# Where are the aspects?



# Late Aspect Identification



☞ Missing *early aspects*.

☞ Difficult to model aspects.

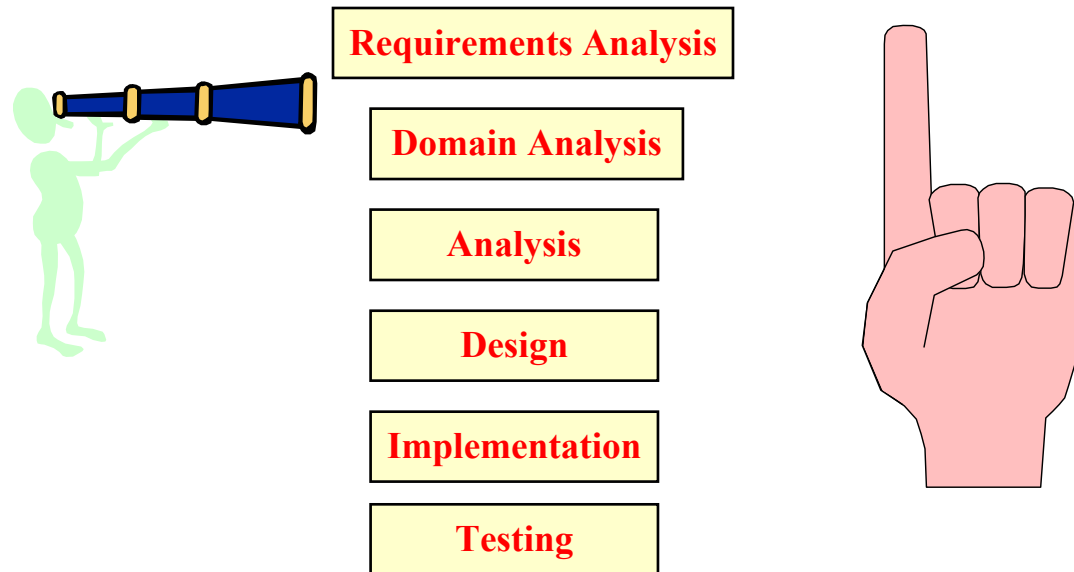
# Modeling Aspects

All right, tracing is an aspect  
But what is tracing/  
how to model this?



```
class Point {  
    void setX(int x) {  
  
        DisplayTracker.updatePoint(this);  
  
        Tracer.traceEntry("Point.set");  
        _x = x;  
        Tracer.traceExit("Point.set");  
    }  
}
```

# Early Aspect Identification



 **All relevant aspects can be identified.**

 **Modeling aspects from domain knowledge.**

# What is a method?

## Webster's dictionary:

- **Method:** 1: a procedure or process for attaining an object: as **a** (1): a systematic procedure, technique, or mode of inquiry employed by or proper to a particular discipline or art (2): a systematic plan followed in presenting material for instruction **b** (1): a way, technique, or process of or for doing something (2): a body of skills or techniques.
- **Methodology.** A branch of philosophy dealing with the science of method; a system of methods and rules applied in a science.

# Rationale for method

- Guidelines for production
- Guidelines for verification
- Provides logical consistency among the different processes and phases in design.
- Helps to reduce possible errors
- Helps to identify important progress milestones.



# Aspect-Oriented Method

- Identifying
- Specifying
- Evaluating aspects
  
- in a systematic way

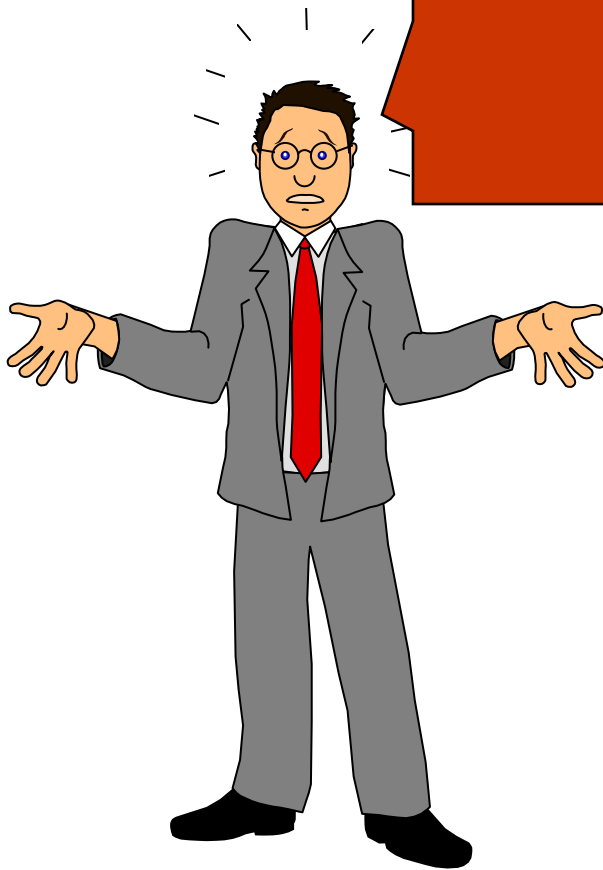
# Rationale for aspect-oriented method

- Guidelines for identification/production of aspects
- Guidelines for verification of aspects
- Provides logical consistency among the different processes and phases in design; consistency for/with aspects.
- Helps to reduce possible errors in aspect-oriented programs/designs
- Helps to identify important progress milestones.



# Architecture Design Methods

Ok, but is there a method to identify aspects?

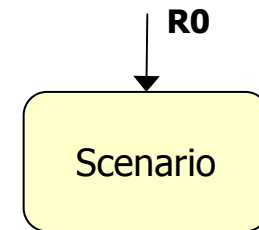


## ASAAM: Aspectual Software Architecture Analysis Method

1. Describe candidate architecture
2. Develop and prioritize scenarios
3. Individual scenario evaluation and aspect identification
  - Direct Scenarios, Indirect Scenarios, Aspectual scenarios, Architectural aspects
4. Scenario interaction evaluation and component classification
  - Cohesive component, tangled component, composite component, ill-defined component
5. Refactor architecture
  - using conventional techniques (OO, patterns etc.)
  - using aspect-oriented techniques

# Heuristic rules for scenario evaluation

- **R0:**  
Develop **SCENARIO** artifacts based on **PROBLEM DESCRIPTION**



## Scenarios

S1. Start multiple processes at the same time.  
S2. Change color of widgets in the window.  
S3. Close all open windows  
S4. Change screen resolution  
S5. Enter a command to start application process  
S6. Move the main window  
S7. Screen saver is activated  
S8. Resize a window  
S9. Terminate a process  
S10. Interrupt a process

S11. Change look-and-feel style on run-time.  
S12. Add voice control  
S13. A failure occurs and the system shuts down.  
S14. Provide dual display screen.  
S15. Use multiple desktops.  
S16. Monitor activities of the user  
S17. Provide touch screen and light pen as input  
S18. A memory overflow due to many opened windows  
S19. Port system to command-based operation system  
S20. Minimize windows after idle time

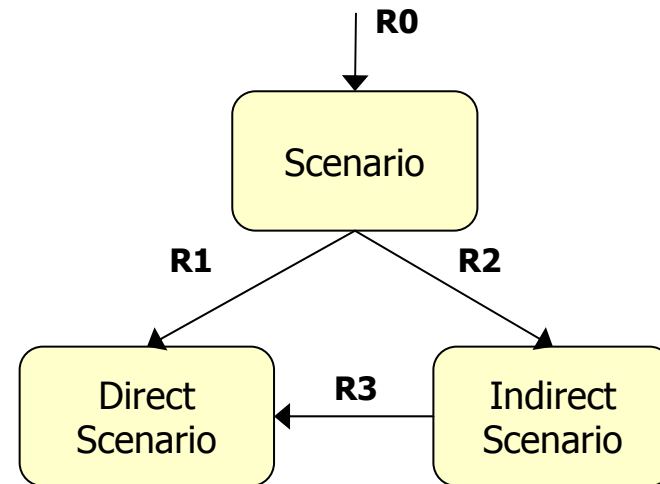
# Extract Scenarios

- Brainstorming Session with different stakeholders
  - Collect Scenarios of different stakeholders
  - Workshop
- Look at domain
  - Possible uses of system
  - Risks
- Look at other systems
  - Compare systems
  - Experiences
- *Reuse* Scenarios
- Evaluate scenarios
- Select scenarios



# Heuristic rules for scenario evaluation

- **R1:**  
**IF SCENARIO** does not require any changes to architectural description  
**THEN SCENARIO** becomes **DIRECT SCENARIO**
- **R2:**  
**IF SCENARIO** requires changes to one or more **ARCHITECTURAL COMPONENTS**  
**THEN SCENARIO** becomes **INDIRECT SCENARIO**
- **R3:**  
**IF INDIRECT SCENARIO** can be resolved after refactoring  
**THEN INDIRECT SCENARIO** is **DIRECT SCENARIO**



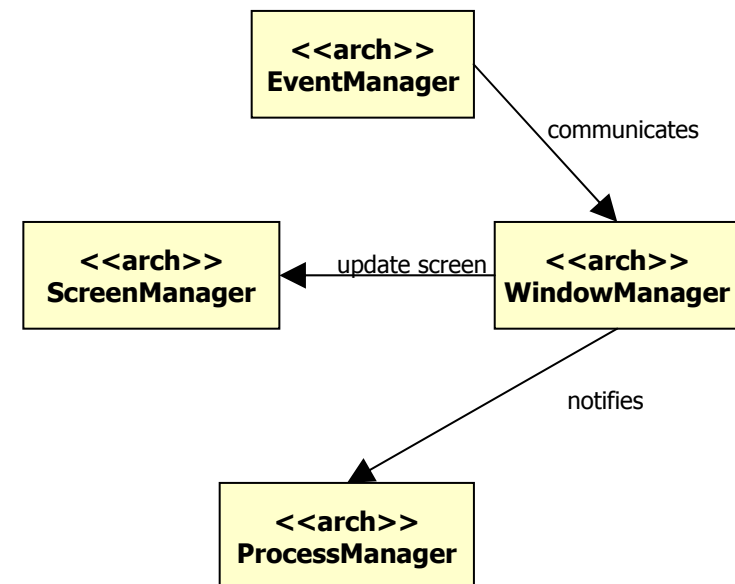
# Direct and Indirect Scenarios

## Direct Scenarios

- S1. Start multiple processes at the same time.
- S2. Change color of widgets in the window.
- S3. Close all open windows
- S4. Change screen resolution
- S5. Enter a command to start application process
- S6. Move the main window
- S7. Screen saver is activated
- S8. Resize a window
- S9. Terminate a process
- S10. Interrupt a process

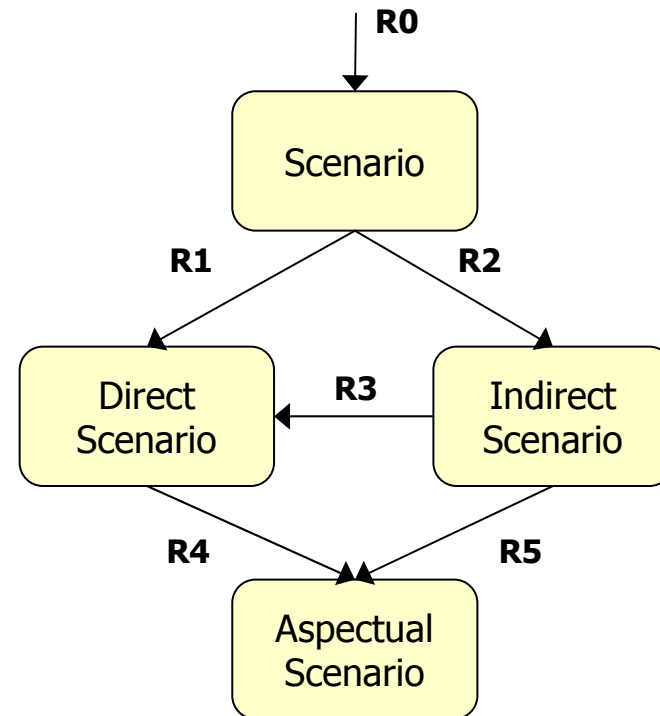
## Indirect Scenarios

- S11. Change look-and-feel style on run-time.
- S12. Add voice control
- S13. A failure occurs and the system shuts down.
- S14. Provide dual display screen.
- S15. Use multiple desktops.
- S16. Monitor activities of the user
- S17. Provide touch screen and light pen as input
- S18. A memory overflow due to many opened windows
- S19. Port system to command-based operation system
- S20. Minimize windows after idle time



# Heuristic rules for scenario evaluation

- **R4:**  
**IF DIRECT SCENARIO** is scattered and cannot be localized in one component  
**THEN DIRECT SCENARIO** is **ASPECTUAL SCENARIO**
- **R5:**  
**IF INDIRECT SCENARIO** is scattered and cannot be localized in one component  
**THEN INDIRECT SCENARIO** is **ASPECTUAL SCENARIO**



# Aspectual Scenarios

## Direct Scenarios

- S1. Start multiple processes at the same time.
- S2. Change color of widgets in the window.
- S3. Close all open windows
- S4. Change screen resolution
- S5. Enter a command to start application process
- S6. Move the main window
- S7. Screen saver is activated
- S8. Resize a window
- S9. Terminate a process
- S10. Interrupt a process

## Aspectual Scenarios

-

## Indirect Scenarios

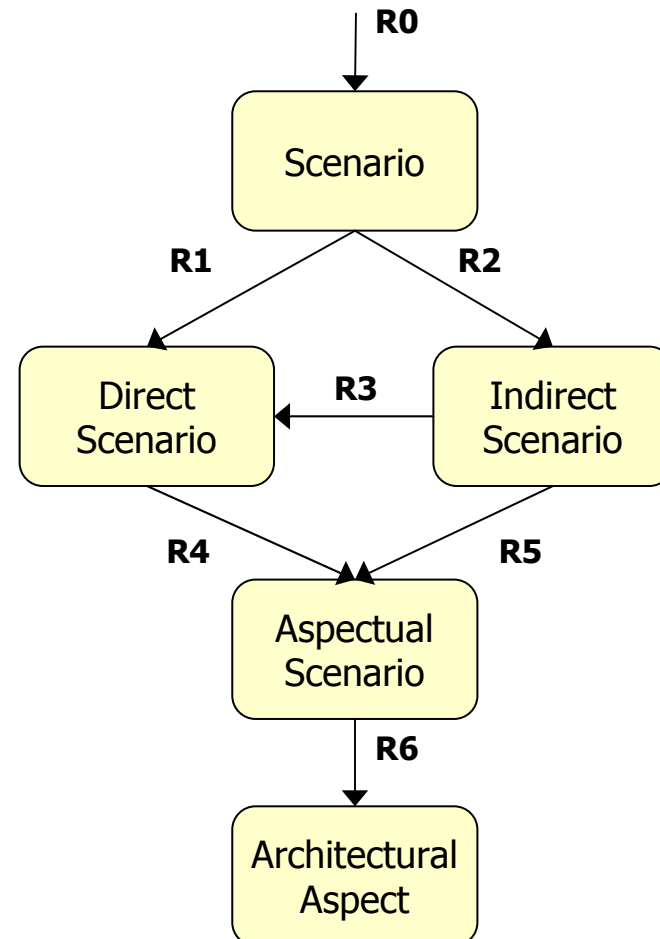
- S11. Change look-and-feel style on run-time.
- S12. Add voice control
- S14. Provide dual display screen.
- S15. Use multiple desktops.
- S17. Provide touch screen and light pen as input
- S18. A memory overflow due to many opened windows
- S20. Minimize windows after idle time for each

## Aspectual Scenarios

- S13. A failure occurs and the system shuts down.
- S16. Monitor activities of the user
- S19. Port system to command-based operation system

# Heuristic rules for scenario evaluation

- **R6:**  
Derive **ARCHITECTURAL ASPECT**  
from **ASPECTUAL SCENARIO**



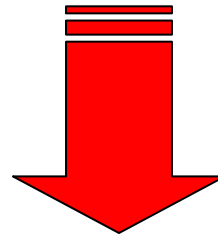
# Architectural Aspects

## Aspectual Scenarios

S13. A failure occurs and the system shuts down.

S16. Monitor activities of the user

S19. Port system to command-based operation system



## Domain Analysis

## Architectural Aspects

Failure Management

Logging

Restarting

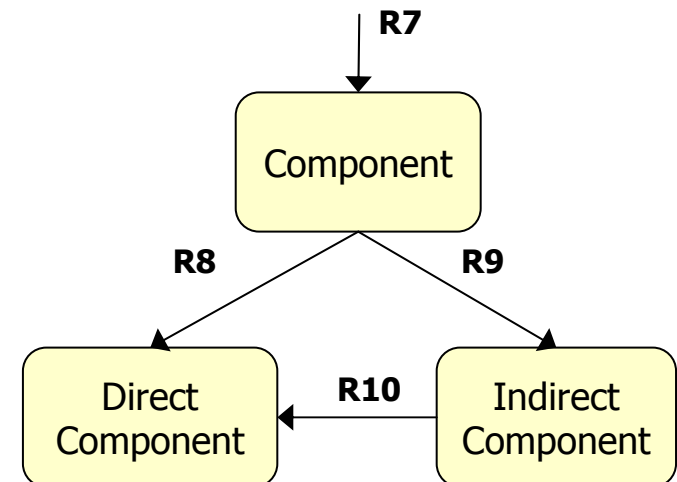
Checkpointing

Monitoring

Operating System Aspect

# Heuristic Rules for Component Classification

- **R7:**  
Select **COMPONENT** from **ARCHITECTURE**
- **R8:**  
**IF COMPONENT** is not affected by a **SCENARIO**  
**THEN** component is **DIRECT COMPONENT** for **SCENARIO**
- **R9:**  
**IF COMPONENT** is affected by a **SCENARIO**  
**THEN** component is **INDIRECT COMPONENT** for **SCENARIO**
- **R10**  
**IF INDIRECT COMPONENT** can be refactored to perform **INDIRECT SCENARIO**  
**THEN INDIRECT COMPONENT** is **DIRECT COMPONENT**

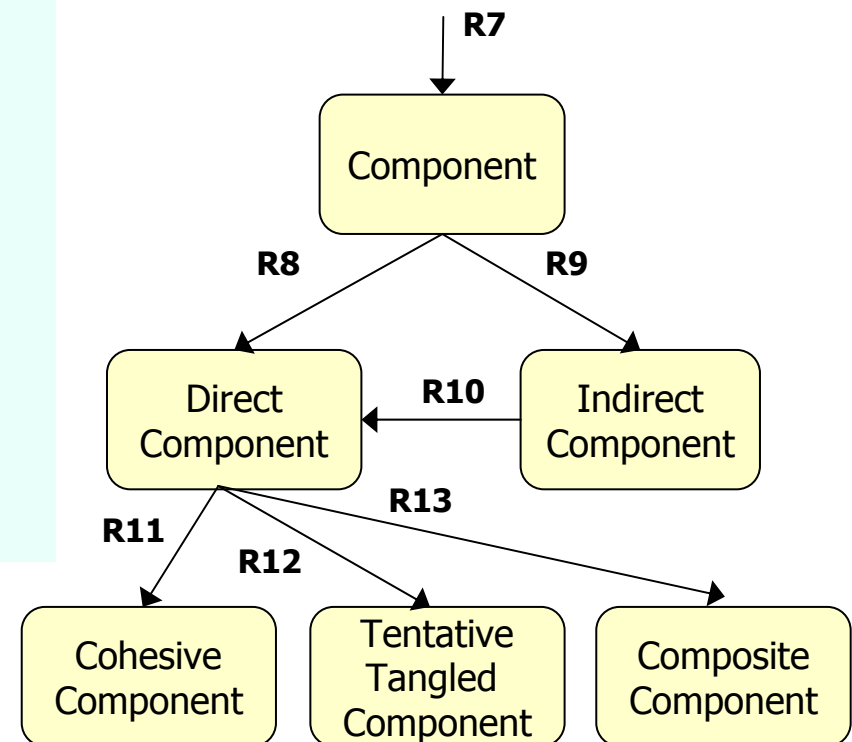


# Direct and Indirect Components

<b>Component</b>	<b>Direct for Scenarios</b>	<b>Indirect for Scenarios</b>
EventManager	S3, S5, S8	S12,S13,S16, S17,S18,S19
ScreenManager	S4, S7	S16,18,S19,S20
WindowManager	S2, S3, S6,S8	S11, S14, S15, S16,S19
ProcessManager	S1, S3, S9, S10	S13, S16,S19

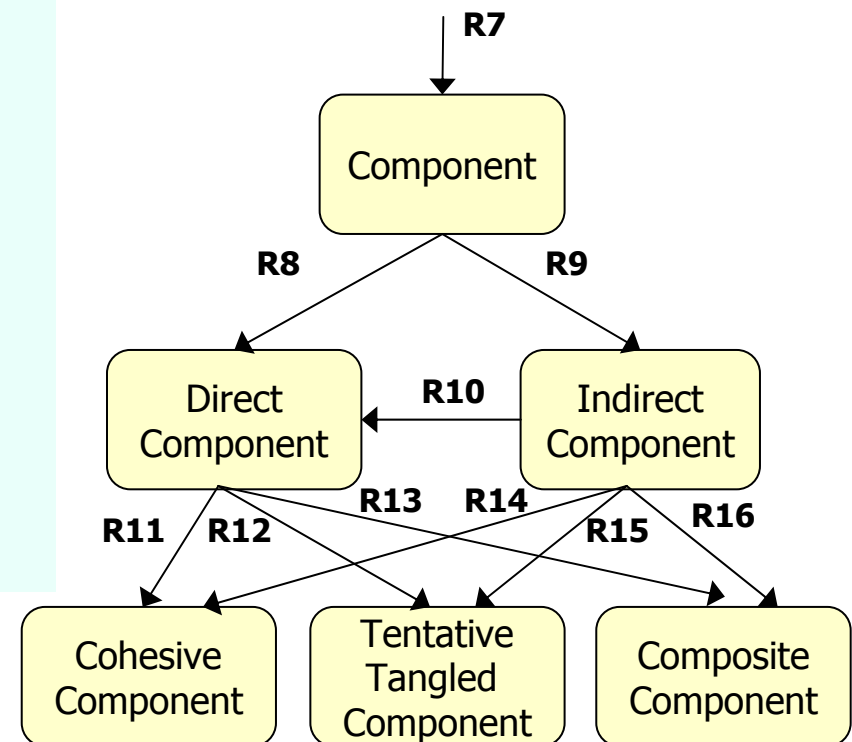
# Cohesive, Tentative Tangled or Composite Component

- **R11**  
**IF DIRECT COMPONENT** performs semantically close scenarios  
**THEN DIRECT COMPONENT** is **COHESIVE COMPONENT**
- **R12**  
**IF DIRECT COMPONENT** performs semantically distinct scenarios **AND** cannot be decomposed  
**THEN DIRECT COMPONENT** is **TENTATIVE TANGLED COMPONENT**
- **R13**  
**IF DIRECT COMPONENT** performs semantically distinct scenarios **AND** can be decomposed  
**THEN DIRECT COMPONENT** is **COMPOSITE COMPONENT**



# Cohesive, Tentative Tangled or Composite Component

- **R14:**  
**IF INDIRECT COMPONENT** includes semantically close scenarios  
**THEN INDIRECT COMPONENT** is **COHESIVE COMPONENT**
- **R15:**  
**IF INDIRECT COMPONENT** includes semantically distinct scenarios **AND** cannot be decomposed  
**THEN COMPONENT** becomes **TENTATIVE TANGLED COMPONENT**
- **R16:**  
**IF INDIRECT COMPONENT** includes semantically distinct scenarios **AND** can be decomposed  
**THEN INDIRECT COMPONENT** is **COMPOSITE COMPONENT**

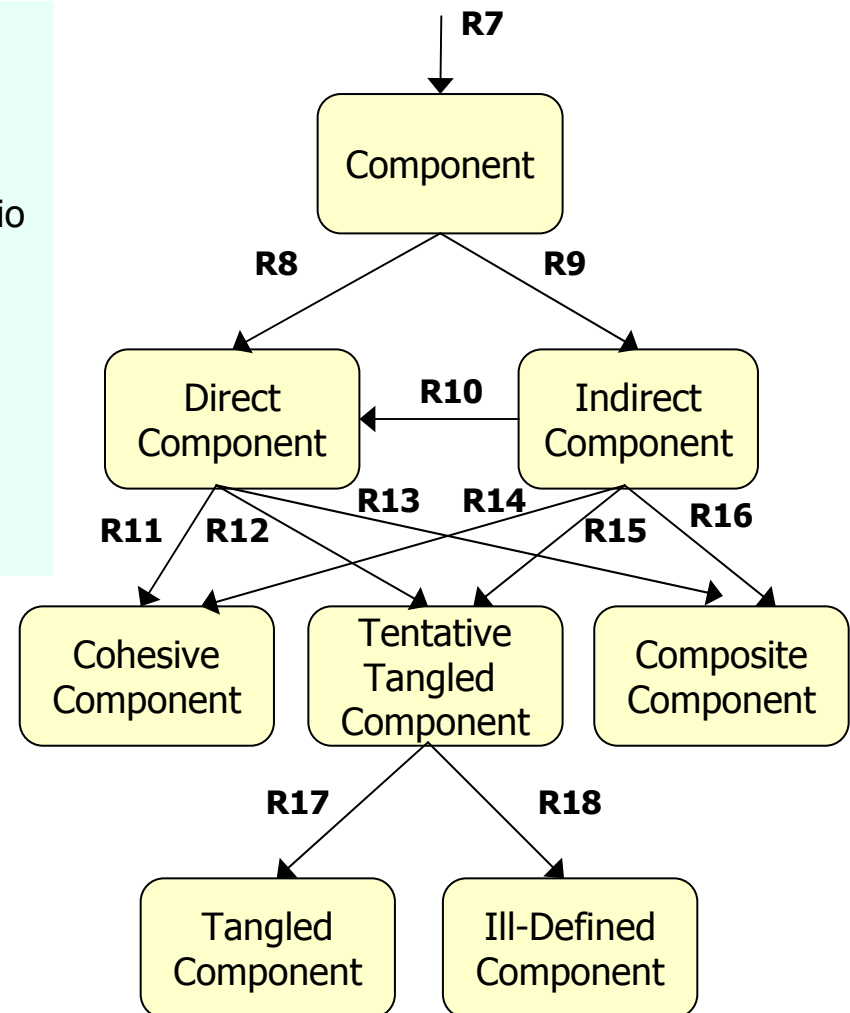


## Cohesive, Tentative Tangled or Composite Component

Component	Cohesive	Tent. Tangled	Composite
EventManager	S3, S5	S13,S16,S19	S12,S17
ScreenManager	S14	S13,S19	S4,S7
WindowManager	S2,S3,S6, S8,S20	S16,S19	S11,S18,S15
ProcessManager	S1,S9,S10	S13,S16,S19	

# Tangled Component or Ill-Defined Component

- **R17:**  
**IF TENTATIVE TANGLED COMPONENT** includes **ASPECTUAL SCENARIO**  
**THEN TENTATIVE TANGLED COMPONENT** is **TANGLED COMPONENT** for given aspectual scenario
- **R18:**  
**IF TENTATIVE TANGLED COMPONENT** does not include **ASPECTUAL SCENARIO**  
**THEN TENTATIVE TANGLED COMPONENT** is **III-DEFINED COMPONENT**



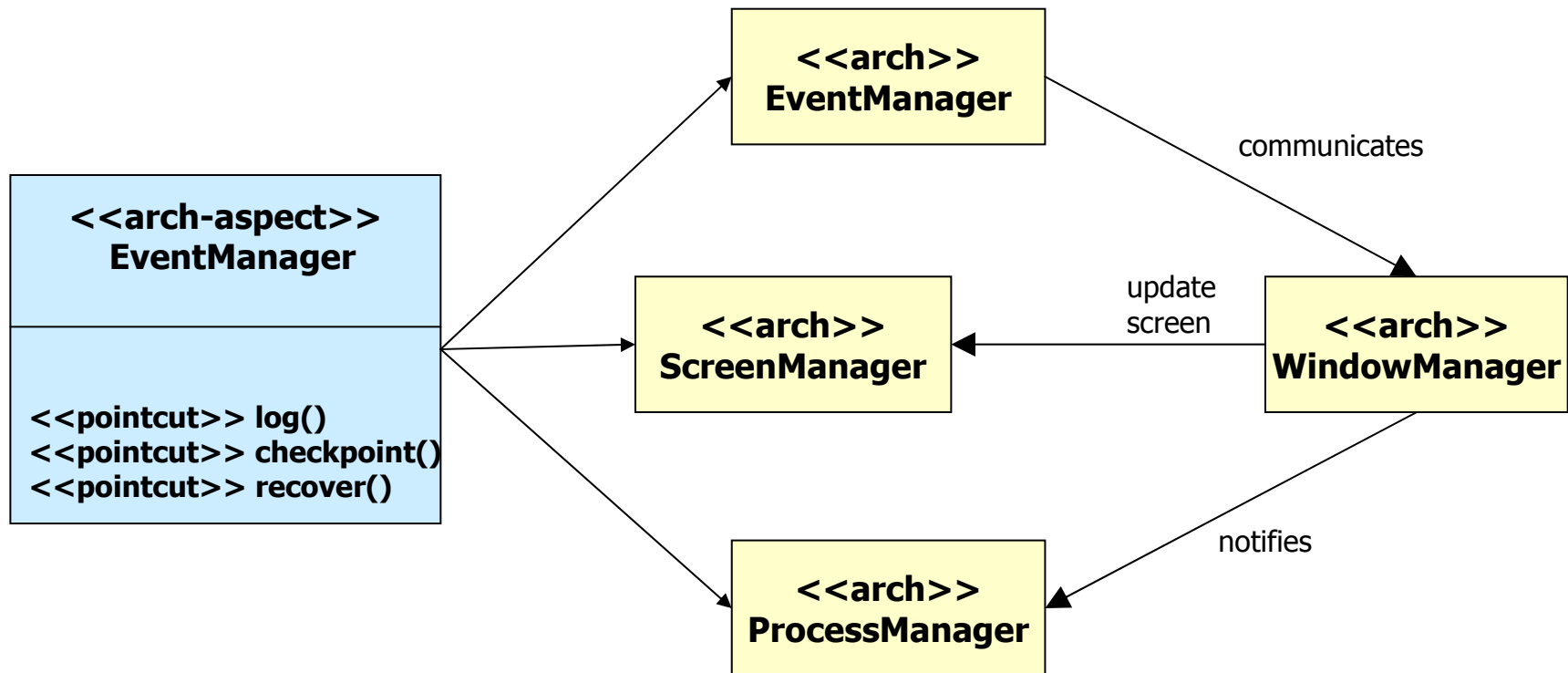
## Identified Aspects and Tangled Components

Component	Cohesive	Tangled (Aspect)	Composite	Ill-defined.
EventManager	S3, S5	S13,S16,S19	S12,S17	-
ScreenManager	S14	S13,S19	S4,S7	-
WindowManager	S2,S3,S6, S8,S20	S16,S19	S11,S18,S15	-
ProcessManager	S1,S9,S10	S13,S16,S19		-

### **Aspects derived from scenarios S13, S16, S19:**

Failure Management,  
Monitoring,  
Operating System Portability

# Aspectual Refactoring



# Conclusion

- Early aspects exist
  - e.g. failure management, monitoring, operating system portability
- ASAAM extends existing scenario based software architecture analysis methods
- to explicitly identify architectural aspects
- and pinpoint aspectual refactoring for corresponding aspects.