# Aspect-Oriented Software Development

## The Role of Early Aspects

**Ana Moreira**

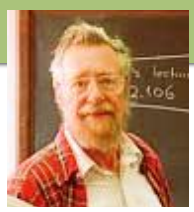
**Departamento de Informática**
**Faculdade de Ciências e Tecnologia**
**Universidade Nova de Lisboa**
*amm@di.fct.unl.pt*
*http://ctp.di.fct.unl.pt/~amm*

**ISCTE, April 15, 2005**

# Separation of Concerns

"This is what I mean by focusing one's attention upon some aspect: it does not mean ignoring the other aspects, it is just doing justice to the fact that from this aspect's point of view, the other is irrelevant." [2]

"Such separation, even if not perfectly possible, is yet the only available technique for effective ordering of one's thoughts that I know of. […] I usually refer to it as '**separation of concerns**'..."[1]



Edsger Dijkstra 1930-2002

[1] E. Dijkstra, *A Discipline of Programming*, Prentice Hall, 1976
[2] E. Dijkstra, "On the role of scientific thought.", 1974

# Modularization

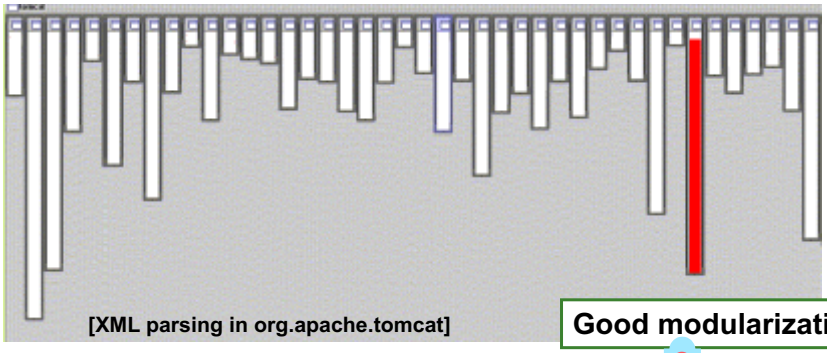
- Information hiding modules (1972)
- Identify design decisions that are likely to change
- Isolate these in separate modules (*separation of concerns*)
- Different design decisions might require different decompositions

D. Parnas, "On the Criteria to Be Used in Decomposing Systems into Modules", CACM 15(12), 1972



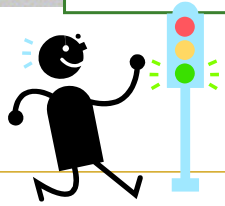


# Modularization: ideal vs real

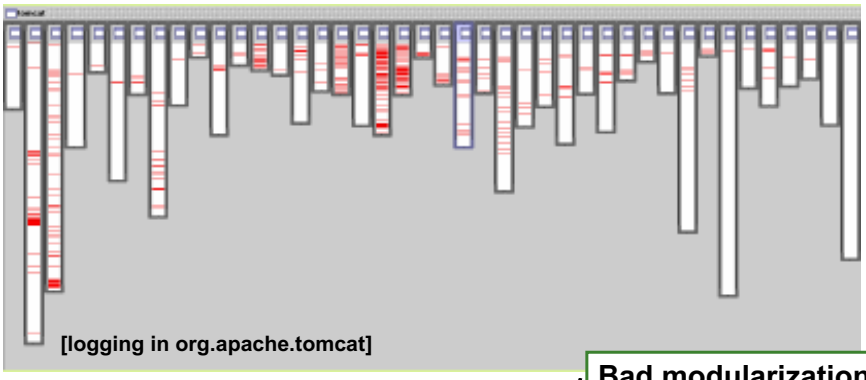


[XML parsing in org.apache.tomcat]

**Good modularization**



# Modularization: ideal vs real



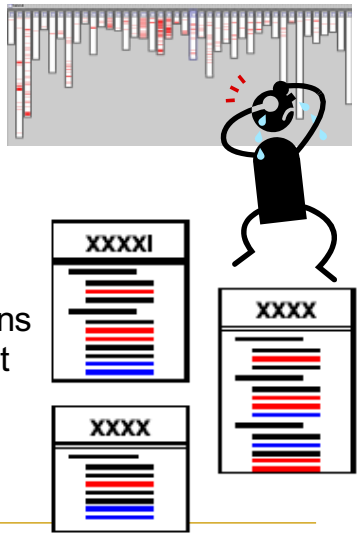

Bad modularization





# Resulting problems

- **Scattered code**
  - The specification of one property is **not encapsulated** in a single module
- **Tangled code**
  - Each module contains descriptions of **several properties** or different functionalities



## Crosscutting concerns (1)

- Separation of concerns that cut across other concerns
  - These *crosscutting* concerns result in systems that are invasive to implement, tough to understand, and difficult to evolve
  - **AOSD aims at providing means for their systematic identification, separation, representation and composition** [Rashid, Moreira, Araujo: AOSD'03]

Encapsulate each (crosscutting) concern in a separate module, the **ASPECT**



## AOSD development stages

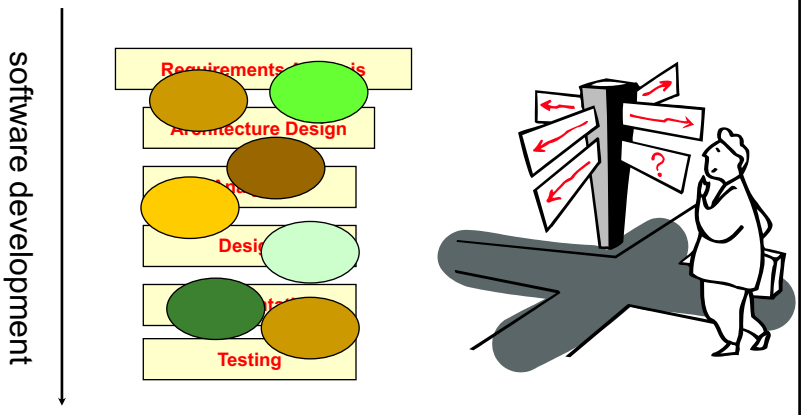
1. **Aspectual decomposition**
   Identify crosscutting concerns
2. **Concern representation**
   Represent each concern *separately*
3. **Aspectual composition (*weaving*)**
   Compose the aspects with other modules

# Aspect-oriented programming mechanisms

- A number of aspect-oriented programming approaches are available:
  - **AspectJ** (1997) [G. Kizales, http://aspectj.org/]
  - **Composition filters** (1991) [Bergmans and Aksit]
  - **DemeterJ/DJ** (1993) [Lieberherr, Orleans, and Ovlinger]
  - **Hyper/J** MDSoC for Java (1999) [Ossher & Tarr]
  - **JAsCo** AOP for component-based SE [http://ssel.vub.ac.be/jasco/index.php]
  - **Aspect Werkz** [Jonas Bonér, http://aspectwerkz.codehaus.org]
  - **Apostle**, Aspect Programming em Smalltalk
  - **AspectC,** uma extensão para C
  - **AspectC++,** uma extensão para C++
  - **JAC, Java Aspect Component** [Pawlak, L. Seinturier, L. Duchien, and G. Florin]
  - …



# But, are aspects only at the implementation level?

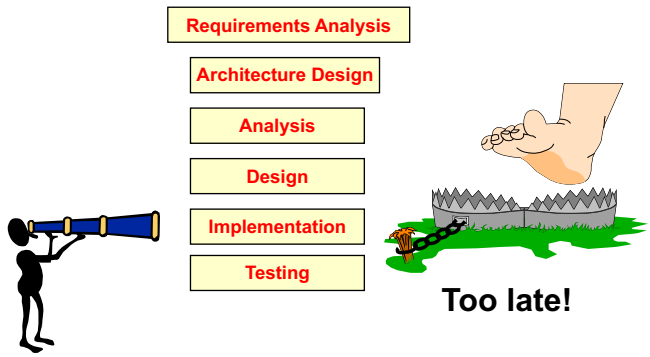


# Aspect-oriented design approaches

- ***Composition Patterns***
  [S. Clarke et al]
- ***Aspect-oriented component engineering***
  [J. Grundy]
- ***Hyperspaces approach***
  [H. Ossher, P. Tarr]
- ***Suzuki and Yamamoto's model***
- ***Separation of Aspects at Design Time***
  [J. Herrero et al]
- ...

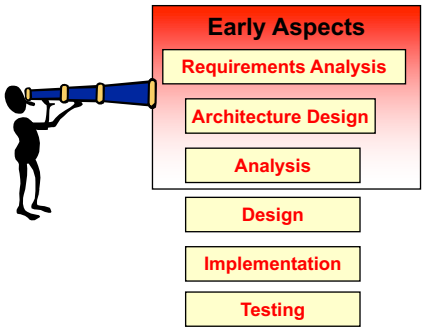



# Late aspect identification




- **Missing *early aspects***
- **Difficult to model aspects**
- **Difficult to trace aspects throughout lifecycle**

# Early aspect identification




- **All relevant aspects can be identified**
- **Modeling aspects from domain knowledge**
- **Improved traceability**



# The Early Aspects movement

- "Early Aspects: Aspect-Oriented Requirements Engineering Architecture Design" at AOSD 2002
  - BCS affiliation
  - http://early-aspects.net/
  - Institutions:
    - Software Engineering Institute, USA (Paul Clements)
    - Lancaster University, UK (Awais Rashid)
    - University of Twente, The Netherlands (Bedir Tekinerdogan)
    - Universidade Nova de Lisboa, Portugal (Ana Moreira & João Araújo)

# Motivations for AORE

1. **Provide support for crosscutting properties during RE**
   - Hence offering a better means to identify and manage conflicts arising due to tangled representations
2. **Identify the mapping and influence on artefacts at later development stages**
   - Hence establishing critical trade-offs before the architecture is derived




# AORE Approaches

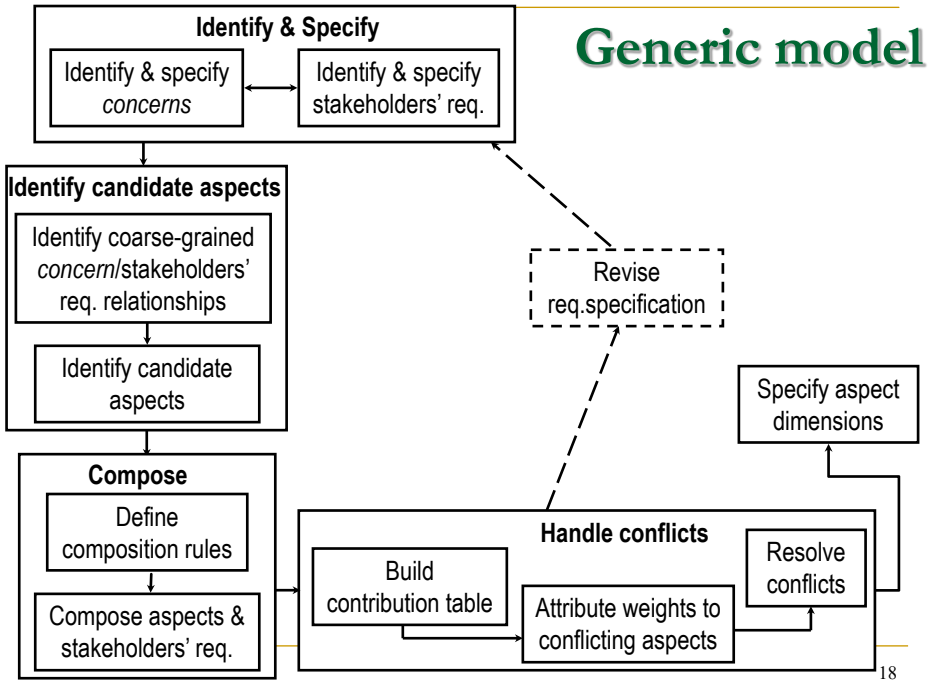
- AORE [Grundy]
  - Targeted to CBSE
  - Provided and required aspects
  - Aspect identification for each component not clearly defined
- AORE [UNL, Lancaster]
  - ARCaDe, two-dimensional approach [Rashid, Moreira, Araújo]
  - AORE, PhD work [Brito, Moreira]
  - MD-AORE, multidimensional approach [Moreira, Araújo, Rashid]

# Modularization and Composition of Aspectual Requirements

The approached discussed next results from the work with Awais Rashid (Univ. of Lancaster) and João Araújo (FCT UNL), particularly the paper "**Modularisation and composition of aspectual requirements**", published in the International ACM conference on Aspect-Oriented Software Development (AOSD) in 2003




# Generic model

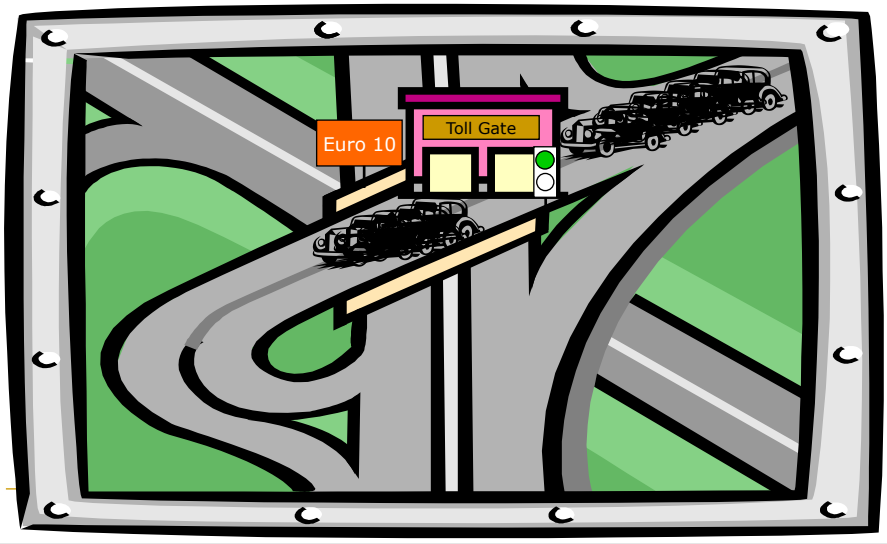


# Concrete Instantiation of the Model

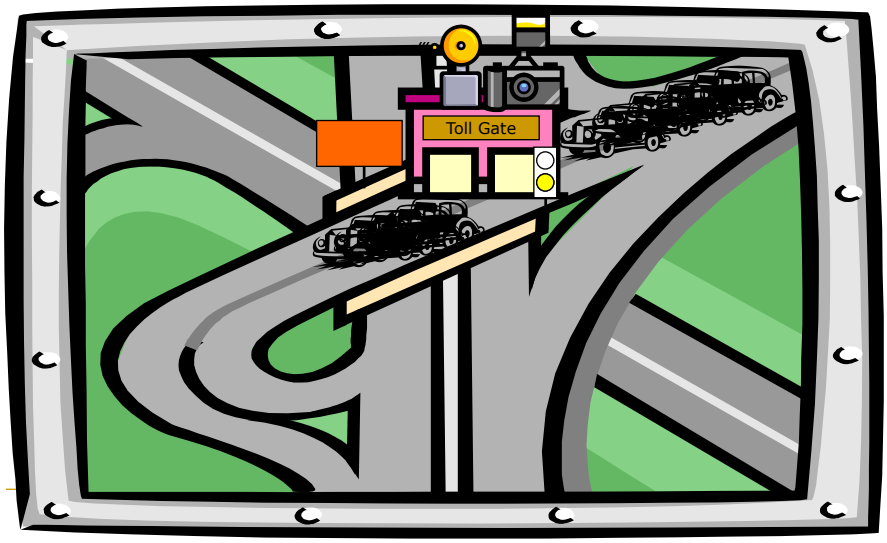
- Viewpoints
  - Specify stakeholder requirements
- *Concerns*
  - Broadly scoped properties
- XML
  - Extensible language for specification of viewpoints, candidate aspects and their composition
- ARCaDe: Aspectual Requirements Composition and Decision support tool
  - DOM, SAX and eXist Native XML database

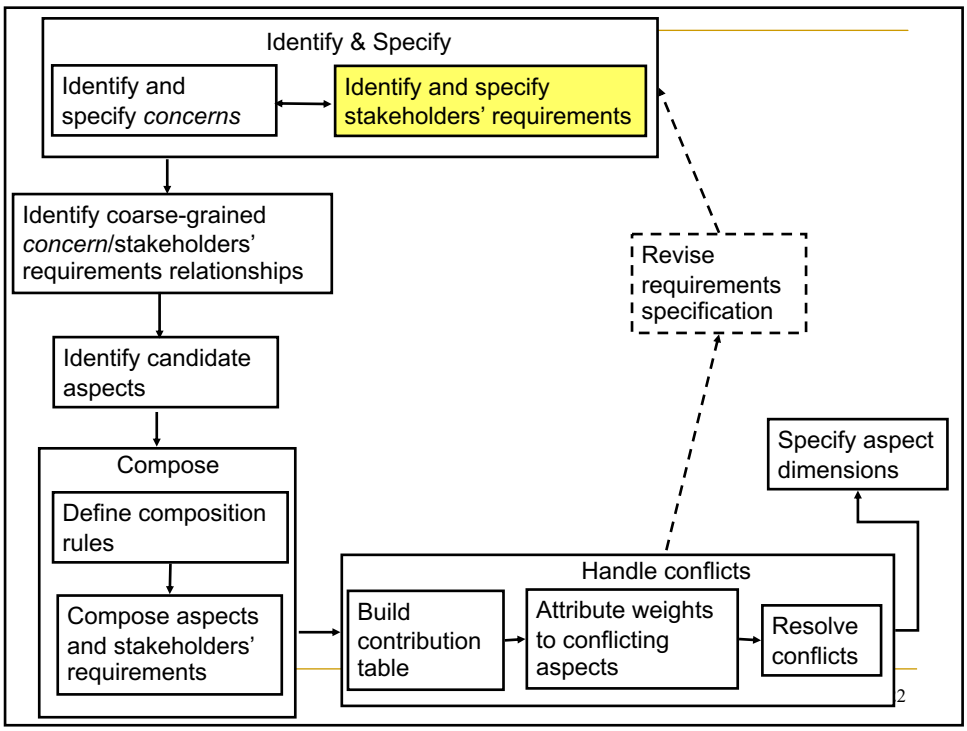



# Toll collection system: **authorised** vehicle



# Toll collection system: **unauthorised** vehicle



# Viewpoints (and sub-viewpoints)

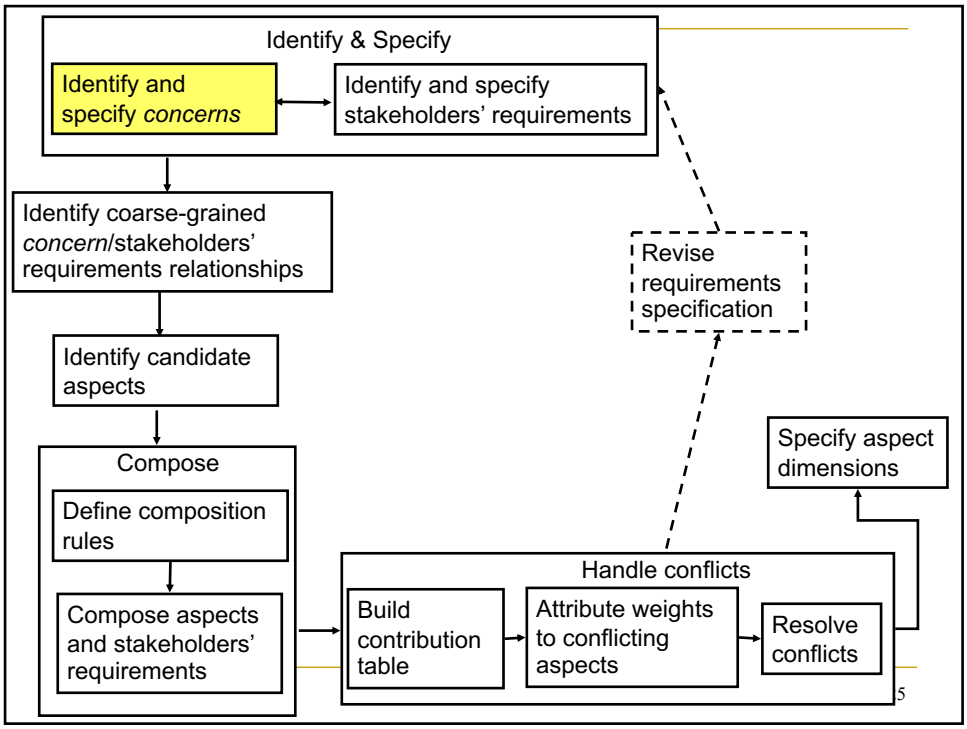
- ATM
- Vehicle
  - Unauthorised Vehicle
- Gizmo
- Toll Gate
  - Entry Toll
  - Paying Toll
    - Single Toll
    - Exit Toll
- Police
- Debiting System
- System Administrator
- Vehicle Owner
  - Registration
  - Billing
  - Activation




# Vehicle Viewpoint

```
<?xml version="1.0" ?>
- <Viewpoint name="Vehicle">
    <Requirement id="1">The vehicle enters the system when it is within
      ten meters of the toll gate. </Requirement>
    <Requirement id="2">The vehicle enters the toll gate.</Requirement>
    <Requirement id="3">The vehicle leaves the toll gate.</Requirement>
    <Requirement id="4">The vehicle leaves the system when it is twenty
      meters away from the toll gate.</Requirement>
      - <Viewpoint name="UnauthorisedVehicle">
          <Requirement id="1">The vehicle number plate will be
            photographed.</Requirement>
      </Viewpoint>
</Viewpoint>
```

**Subviewpoint**




## Concerns

- Security
- Response Time
- Multi-Access
- Compatibility
- Legal Issues
- Correctness
- Availability

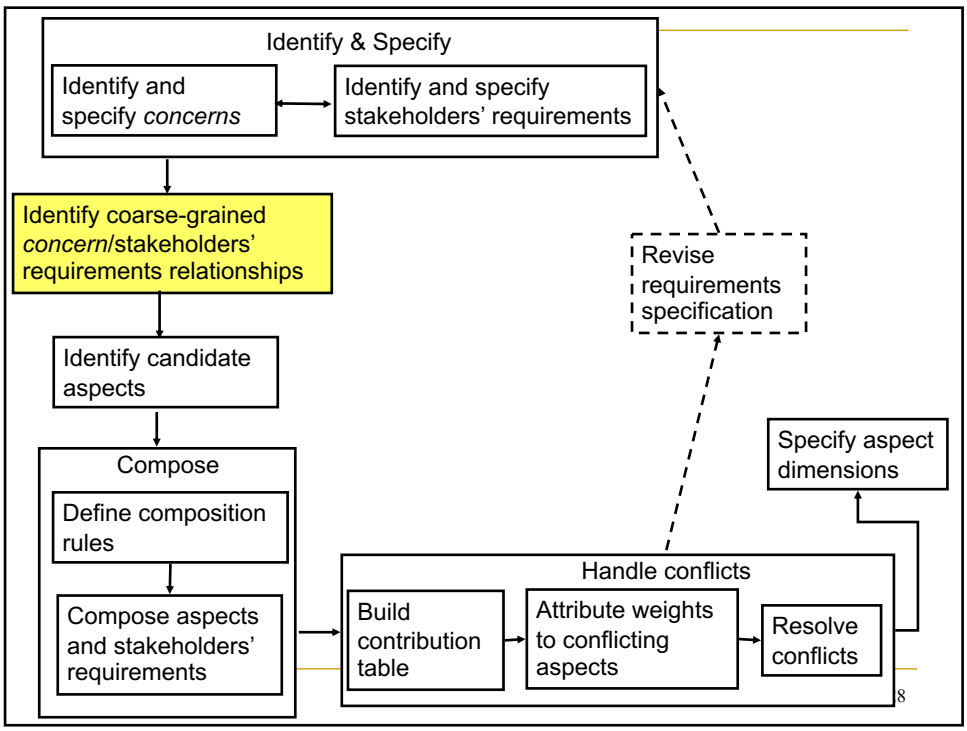
# Response Time *concern*

```
<?xml version="1.0" ?>
- <Concern name="ResponseTime">
    - <Requirement id="1"> The system needs to react in-time in order to:
        <Requirement id="1.1">read the gizmo identifier; </Requirement>
        <Requirement id="1.2">turn on the light (to green or
                              yellow);</Requirement>
        <Requirement id="1.3">display the amount to be paid;</Requirement>
        <Requirement id="1.4">photograph the plate number from the
                              rear;</Requirement>
        <Requirement id="1.5">sound the alarm;</Requirement>
        <Requirement id="1.6">respond to gizmo activation and
                              reactivation.</Requirement>
    </Requirement>
  </Concern>
```

**Subrequirements**







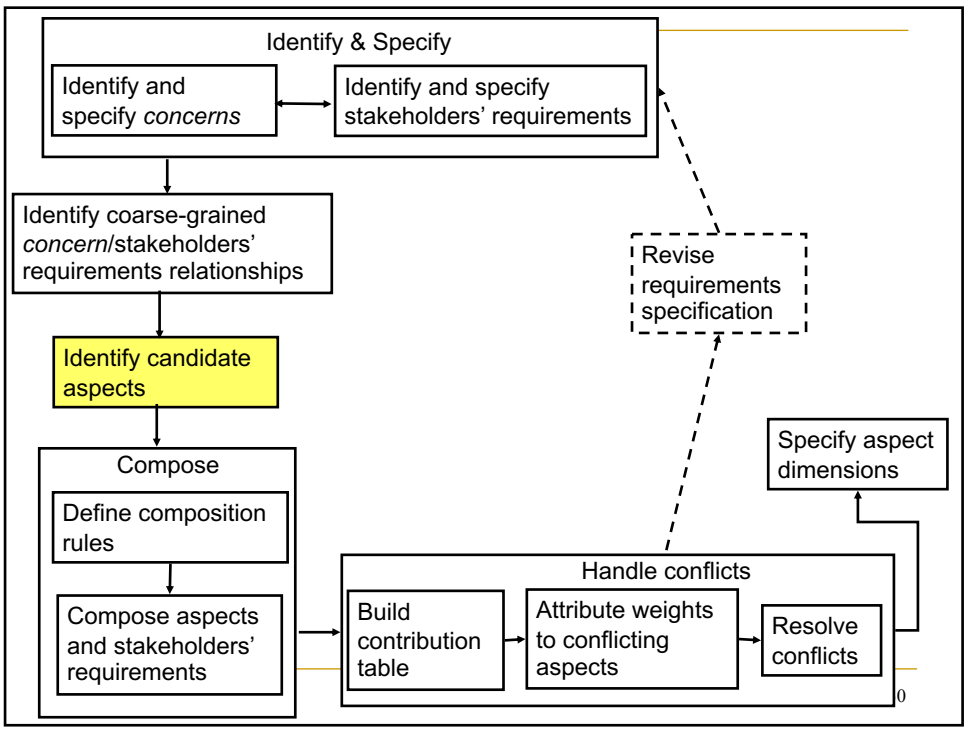
# Identify Coarse-grained concerns/viewpoint relationships

| Concerns \ VP | Pol | Gz | DS | ATM | TG | ... | Vh | UV | VO | ... | Adm |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Response Time | | ✓ | | ✓ | ✓ | | ✓ | ✓ | | | |
| Availability | | ✓ | | ✓ | ✓ | | | | | | ✓ |
| Security | ✓ | | ✓ | ✓ | | | | | ✓ | | ✓ |
| Legal Issues | ✓ | | | | | | | | | | |
| Compatibility | ✓ | | ✓ | ✓ | | | | | | | |
| Correctness | ✓ | ✓ | ✓ | | ✓ | | | | ✓ | ✓ | |
| Multi Access | | ✓ | | ✓ | ✓ | | ✓ | ✓ | | | ✓ |

**Legend:** Pol: Police; Gz: Gizmo; DS: Debiting System; TG: Toll Gate; Vh: Vehicle; UV: Unauthorised Vehicle; VO: Vehicle Owner; Adm: Administrator.



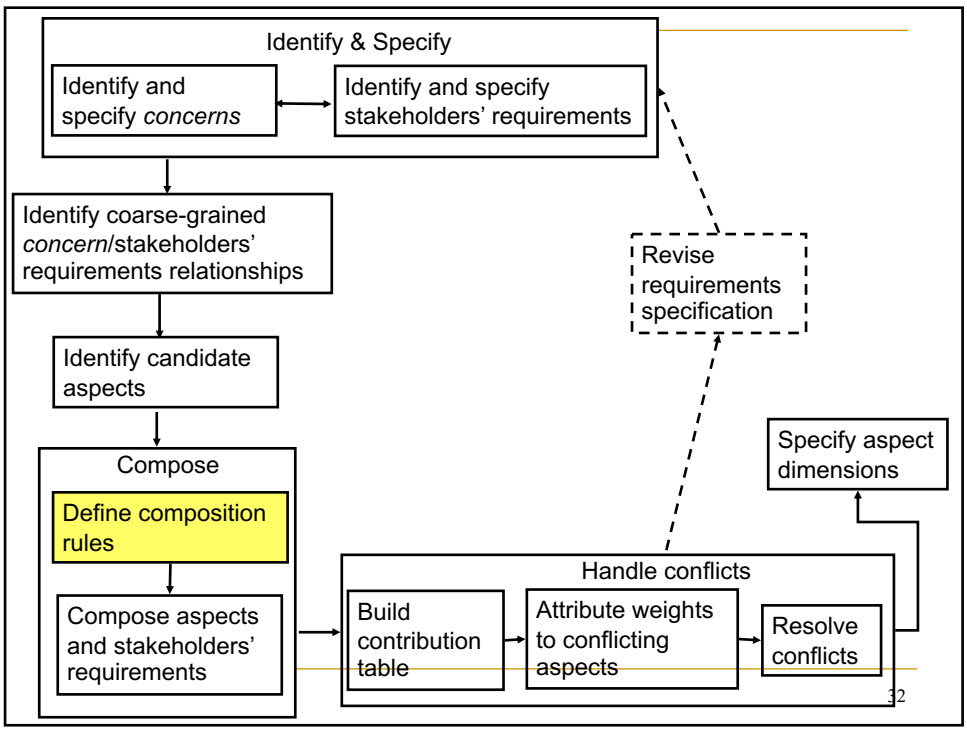




# Identify Candidate Aspects

- *Concerns* influencing and constraining multiple viewpoints
  - **Response Time:** Gizmo, ATM, Toll Gate, Vehicle
  - **Compatibility:** Police, Debiting System, ATM
- Transform XML definition of *concern*
  - Replace <concern> tag with <aspect> tag
    - Simple XSLT transformation
  - Reflect the aspectual nature of the *concern*



# Composition rules for Response Time

Action and operator specifying how the viewpoint requirements are constrained by the specific aspectual requirements

Sub-requirements must be explicitly excluded or included

The action value describes whether another (or a set of) viewpoint requirement must be *satisfied* or the constraint specified merely *fulfilled.*

```
<?xml version="1.0" ?>
- <Composition>
  - <Requirement aspect="ResponseTime" id="1.1">
    - <Constraint action="enforce" operator="between">
        <Requirement viewpoint="Vehicle" id="1" />
        <Requirement viewpoint="Vehicle" id="2" />
      </Constraint>
    - <Outcome action="satisfied">
        <Requirement viewpoint="Gizmo" id="1" children="include" />
      </Outcome>
    </Requirement>

  - <Requirement asp
    - <Constraint ac
        <Requireme
        <Requireme
      </Constraint>
    - <Outcome action="satisfied" operator="XOR">
        <Requirement viewpoint="PayingToll" id="1" />
        <Requirement viewpoint="PayingToll" id="2" />
      </Outcome>
    </Requirement>
...
</Composition>
```

# Composition Semantics: Constraint Actions

| Constraint Action | | Aspects applicable to |
|---|---|---|
| *Type* | *Description* | |
| **enforce** | Imposes an additional condition over a set of viewpoint reqs. | Response Time |
| **ensure** | States that a condition that should exist for a set of viewpoint reqs. actually exists. | Availability, Compatibility, Correctness |
| **provide** | Specifies additional features to be incorporated for a set of viewpoint reqs. | Security, Multiple Access |
| **applied** | Describes rules that apply to a set of viewpoint reqs. & might alter their outcome. | Legal Issues |
| **exclude** | Exclude some viewpoints or reqs. if the value *all* is specified. | ANY |

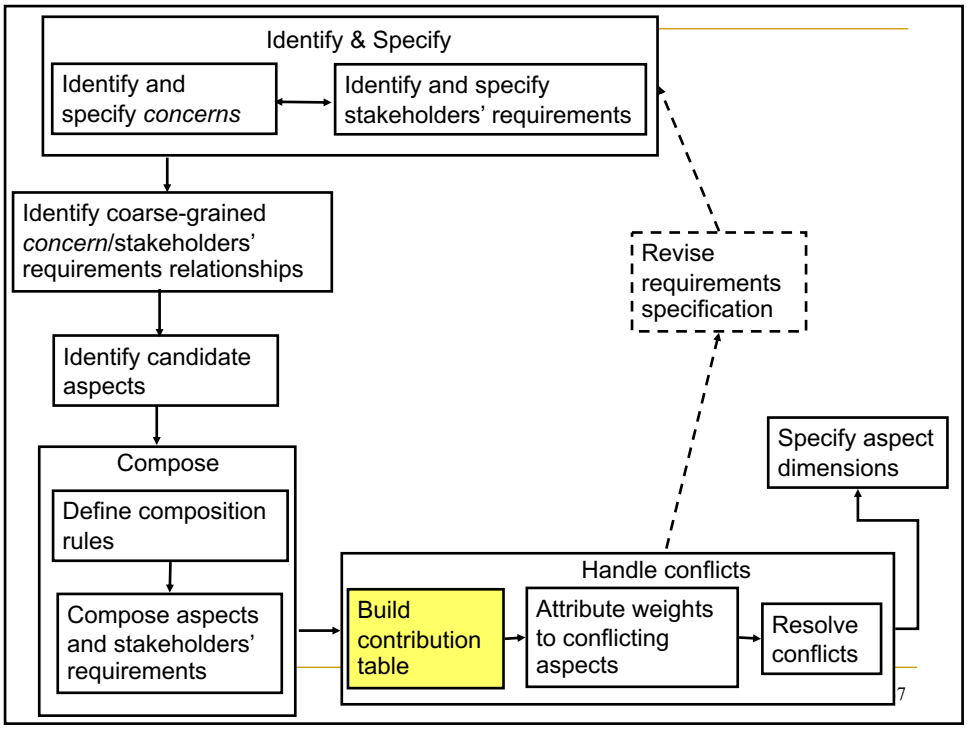
# Composition Semantics: Constraint Operators

| Constraint Operator | | Action | Valid aspect: action-operator combinations |
|---|---|---|---|
| *Type* | *Description* | | |
| **during** | Describes the temporal interval during which a set of reqs. is being satisfied. | **ensure** | Availability: ensure-during |
| **between** | Describes the temporal interval falling between the satisfaction of two reqs. | **enforce** | Response Time: enforce-between |
| **on** | Describes the temporal point after a set of reqs. has been satisfied. | **enforce** | Response-Time: enforce-on |
| **for** | Describes that additional features will complement the viewpoint reqs. | **applied, provide** | Legal Issues: applied-for<br>Security: provide-for<br>Multiple Access: provide-for |
| **with** | Describes that a condition will hold for two sets of reqs. with respect to each other. | **ensure** | Compatability: ensure-with |
| **in** | Describes that a condition will hold for a set of reqs. that has been satisfied. | **ensure** | Correctness: ensure-in |
| **XOR** | Exclusive-OR (either req. but not both) | **ANY** | ANY |




# Composition Semantics: Outcome Actions

| Outcome Action | | Aspects applicable to |
|---|---|---|
| *Type* | *Description* | |
| **satisfied** | Asserts that a set of viewpoint reqs will be satisfied after the constraints of an aspectual req. have been applied. | ANY |
| **fulfilled** | Asserts that the constraints of an aspectual req. have been successfully imposed. | ANY |

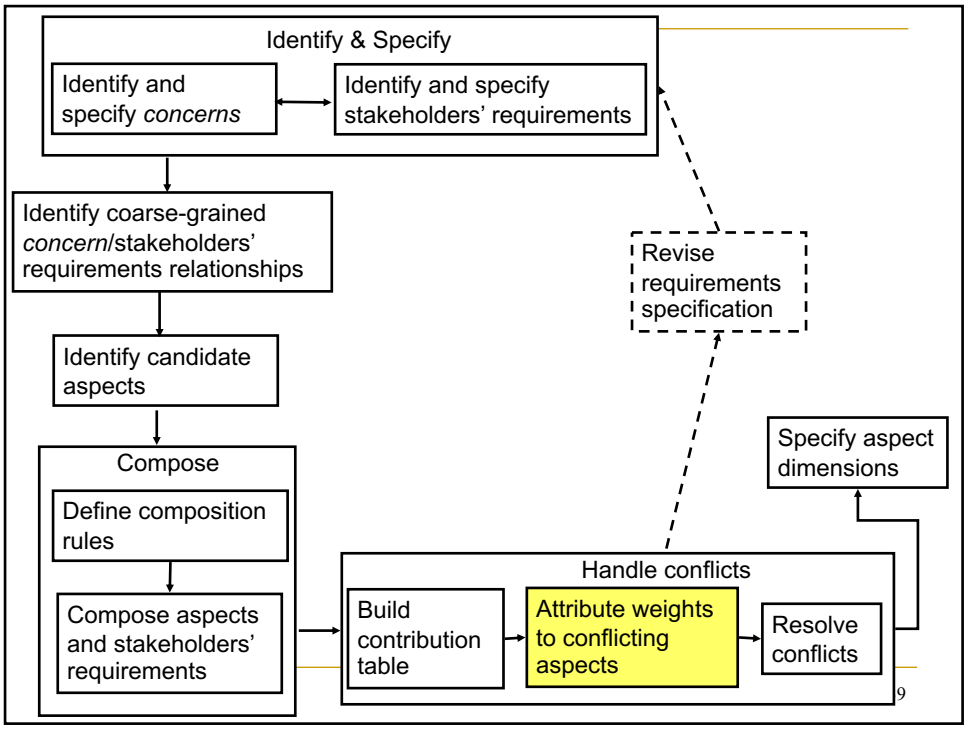



# Build Contribution Table

| Aspects / Aspects | Response Time | Availability | Security | Legal Issues | Compatibility | Correctness | Multi-Access |
|---|---|---|---|---|---|---|---|
| Response Time | | + | − | | | − | − |
| Availability | | | | | | | + |
| Security | | | | | | + | |
| Legal Issues | | | | | + | + | |
| Compatibility | | | | | | | |
| Correctness | | | | | | | |
| Multi-Access | | | | | | | |



38




# Attribute Weights to Conflicting Aspects

- Extent to which an aspect may constrain a viewpoint

*Very important* takes values in the interval ]0,8 .. 1,0]

*Important* takes values in the interval ]0,5 .. 0,8]

*Average* takes values in the interval ]0,3 .. 0,5]

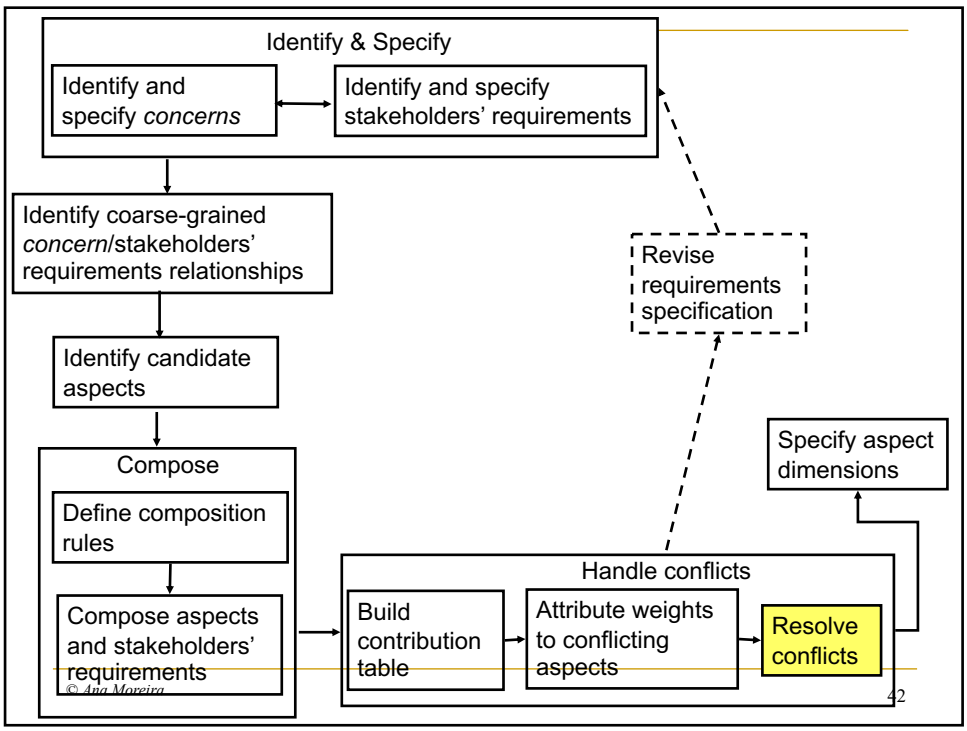
*Not so important* takes values in the interval ]0,1 .. 0,3]

*Do not care much* takes values in the interval [0 .. 0,1]

# Contribution Table with Weights

| *Aspects* \ *VP* | Police | Gizmo | DebSys | ATM | TollGate | PT | ST | ExT | ET | Vehicle |
|---|---|---|---|---|---|---|---|---|---|---|
| Response Time | | 1,0 | | 0,3 | 1,0 | 1,0 | 1,0 | 1,0 | 1,0 | 1,0 |
| Availability | | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| Security | ✓ | | ✓ | 1,0 | | | | | | |
| Legal Issues | ✓ | | | | | | | ✓ | | |
| Compatibility | ✓ | | ✓ | ✓ | | | | | | |
| Correctness | | 0,8 | | | 1,0 | 1,0 | 1,0 | 1,0 | 1,0 | |
| Multi-Access | | 0,3 | | 0,3 | 0,3 | 0,3 | 0,3 | 0,3 | 0,3 | 0,3 |



# Resolve conflicts

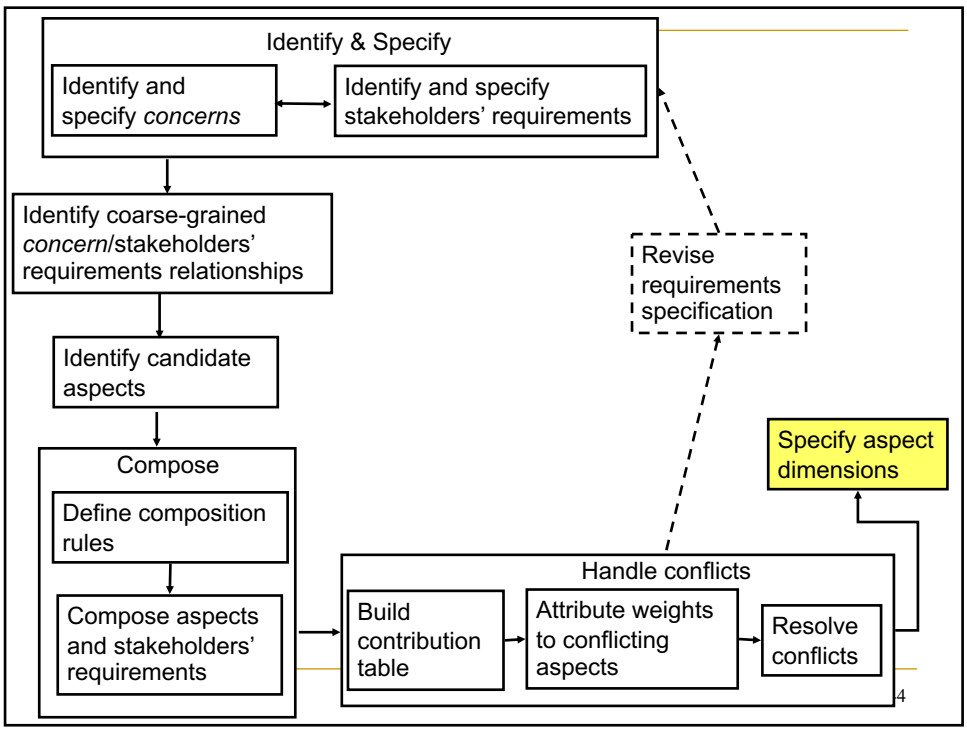
| *VP* / *Aspects* | Police | Gizmo | DebSys | ATM | TollGate | PT | ST | ExT | ET | Vehicle |
|---|---|---|---|---|---|---|---|---|---|---|
| Response Time | | 1,0 | | | 1,0 | 1,0 | 1,0 | 1,0 | 1,0 | 1,0 |
| Availability | | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| Security | ✓ | | ✓ | 1,0 | | | | | | |
| Legal Issues | ✓ | | | | | | | ✓ | | |
| Compatibility | ✓ | | ✓ | ✓ | | | | | | |
| Correctness | | 0,8 | | | 1,0 | 1,0 | 1,0 | 1,0 | 1,0 | |
| Multi-Access | | 0,3 | | | 0,3 | 0,3 | 0,3 | 0,3 | 0,3 | 0,3 |

Need help!

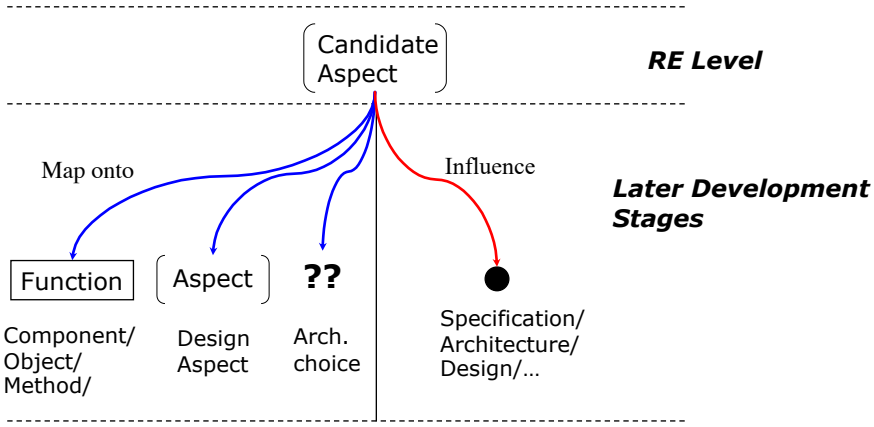
Easy!







# RE aspect dimensions






# Aspect Mapping and Influence

| Candidate aspect | Influence | Mapping |
|---|---|---|
| Compatibility | Specification, architecture,design, evolution | Function |
| Response time | Architecture, design | Aspect |
| Legal issues | Specification | Function |
| Correctness | Specification, design | Function |
| Security | Architecture, design | Aspect |
| Availability | Architecture | Decision |
| Multi-user system | Architecture, design | Aspect |

## Conclusions

- Crosscutting concerns are typically scattered over several modules and result in tangled code.
- AOSD provides explicit abstraction mechanisms to represent these so-called aspects and compose these into programs
- AORE offers better means to identify and manage conflicts arising due to tangled representations
- AORE helps establishing critical trade-offs before the architecture is derived




## Further reading


- A. Moreira, A. Rashid, J. Araujo, **Multi-dimensional Separation of Concerns in Requirements Engineering**, Proc. RE Conference 2005, IEEE CS Press.: 285-296
- A. Rashid, A. Moreira, J. Araújo, **Modularisation and Composition of Aspectual Requirements**, Proc. AOSD Conference 2003, ACM: 11-20
- I. Brito, A. Moreira, **Advanced Separation of Concerns for Requirements Engineering**, Proc. JISBD Conference 2003: 47-56
- A. Moreira, J. Araújo, I. Brito, **Crosscutting quality attributes for requirements engineering**, Proc. SEKE Conference 2002: 167-174
- Early Aspects Portal: *http://www.early-aspects.net*

## Acknowledgements


- A special word of thanks is due to many people I've worked with, in particular (in alphabetical order):
  - Alessandro Garcia
  - Awais Rashid
  - Bedir Tekinerdogan
  - Elisa Baniassad
  - Isabel Brito
  - Paul Clements
  - Pete Sawyer





# Questions?