

Aspect-Oriented Requirements Engineering

Awais Rashid¹, Ana Moreira², Joao Araujo² and
Ruzanna Chitchyan¹

¹*Lancaster University, UK*

²*Universidade Nova de Lisboa, Portugal*

© A. Rashid, A. Moreira, J. Araujo, R. Chitchyan (2006)

Acknowledgements

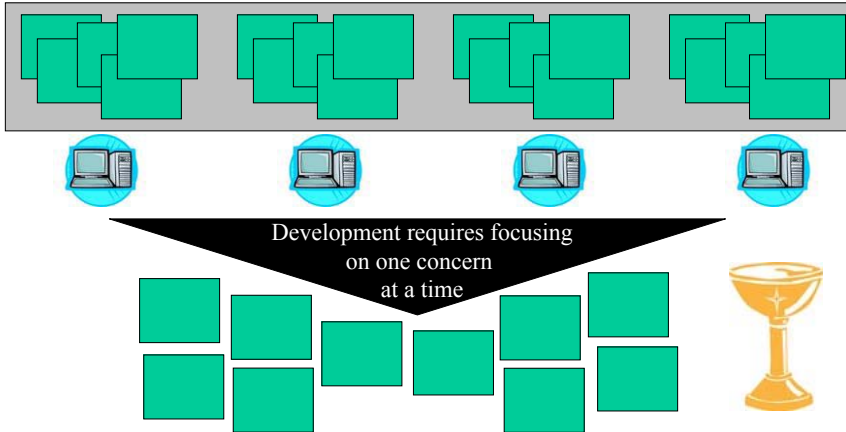
- We have worked with a number of people over the years on this topic. We feel that a special acknowledgement is due to the following people:
 - Alessandro Garcia, Americo Sampaio, Paul Rayson, Peter Sawyer (Lancaster University)
 - Elisa Baniassad (Chinese University of Hong Kong)
 - Isabel Sofia Brito (Universidade Nova de Lisboa)
 - Monica Pinto, Lidia Fuentes (Universidad de Malaga)
 - Thais Batista (UFRN – Brazil)
 - Shmuel Katz (Technion-Israel Institute of Technology)
 - Jon Whittle (George Mason University)
- Work supported by projects: SOFTAS, MULDRE and AOSD-Europe

- The need for aspect-oriented requirements engineering
- Introduction to aspect-oriented software development
- Overview of aspect-oriented requirements engineering
 - How does it fit in with a general RE process
- Managing crosscutting in viewpoints-based requirements models using aspects
- Managing crosscutting in scenario-based requirements models using aspects
- Deriving architectural insights from aspects in requirements

The need for aspect-oriented requirements engineering

Need for Separation of Concerns

Large ... complex ... distributed software systems



5

Separation of Concerns

This is what I mean by focusing one's attention upon a certain aspect; it does not mean completely ignoring the other ones, but temporarily forgetting them to the extent that they are irrelevant for the current topic.

Such a 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 **"a separation of concerns" [...].**

E. Dijkstra, *A Discipline of Programming*,
Prentice Hall, 1976, pp. 210



The Problem of Crosscutting Concerns

- Broadly-scoped concerns
 - Distribution, security, real-time constraints, etc.
 - Crosscutting in nature
 - Severely constrain quality attributes and separation of concerns

What are Crosscutting Concerns?

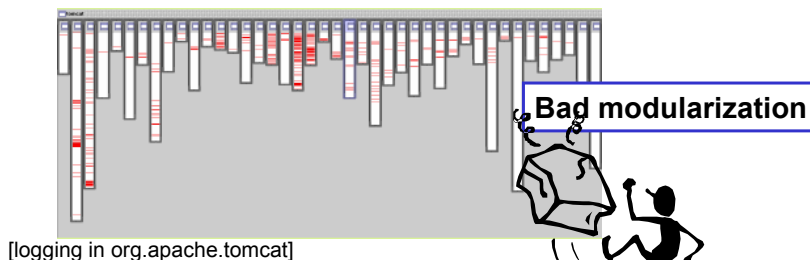
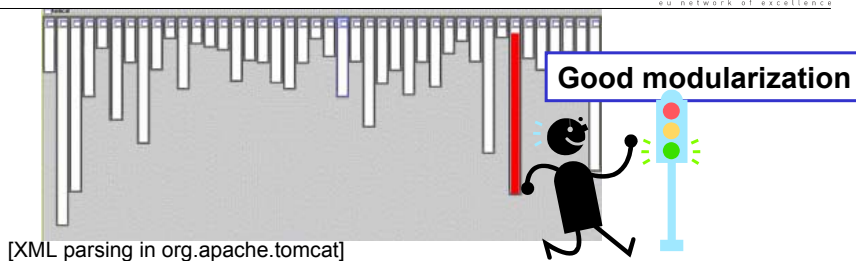
- A concern whose specification (and implementation) is scattered among several other concerns
- A concern that crosscuts various requirements sets or units in the specification
- A broadly scoped property that has an affect on multiple requirements with potential consequences to later development stages

How do Crosscutting Concerns Manifest in Requirements?

- We need to consider the dominant decomposition of those artefacts
 - The dominant decomposition is the **base structure** of the requirements (use case descriptions, viewpoints, etc.)
 - The descriptions of services and constraints that do not fit the decomposition criteria are not localised (are scattered), and the base artefacts have mixed purposes (are tangled)

9

Crosscutting Concerns Affect Modularization

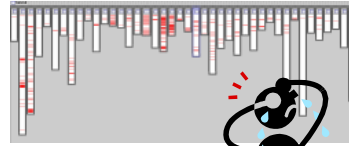


10

Resulting Problems

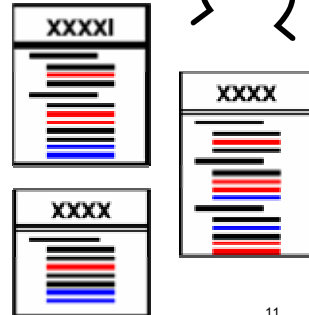
○ Scattering

- The specification of one property is **not encapsulated** in a single requirements unit, e.g., a viewpoint, a use case.



○ Tangling

- Each requirements unit contains descriptions of **several properties** or different functionalities



11

Managing Crosscutting Concerns in Viewpoint Models

- Broadly scoped concerns are identified, but compositional information is embedded in viewpoint definitions (Preview)
- No relationships (crosscutting or others) between viewpoints are identified
- This means that:
 - Compositional information about crosscutting concerns is embedded in the viewpoints
 - Compositional information is limited to references between viewpoints and crosscutting concerns.

12

Motivating Example: Toll Collection System

- In a toll gate system, we *must ensure that the vehicle gizmo is read between the moment the vehicle is detected by the system (when it is within ten meters of the toll gate) and enters the toll gate*
- This requirement involves several concerns: gizmo, vehicle, response time, toll gate.

Name	Toll gate	
Focus	...	
Concerns	response time, security	
Sources	Name	Vehicle
Requirements	Focus	...
History	Concerns	response time

So	Name	Gizmo
Re	Focus	...
His	Concerns	response time

Sources	...
Requirements	
History	

13

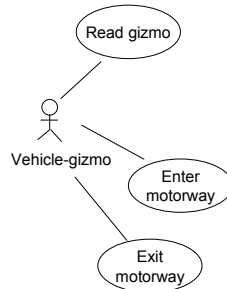
Managing Crosscutting Concerns in Use Case Models

- How to handle broadly-scoped properties using use cases?
 - Typical ones, such as security and response time behaviour, cannot be handled. They do not fit the intrinsic nature of use cases
 - Jacobson and Ng propose “use case slices” to handle this shortcoming
- Can, however, modularize functional concerns using use cases.
 - Crosscutting functional concerns can be captured using the regular use case concepts and relationships, but composition is achieved through a simple relationship
- There is a lack of a fully-fledged, systematic composition mechanism

14

Motivating Example: Toll Collection System

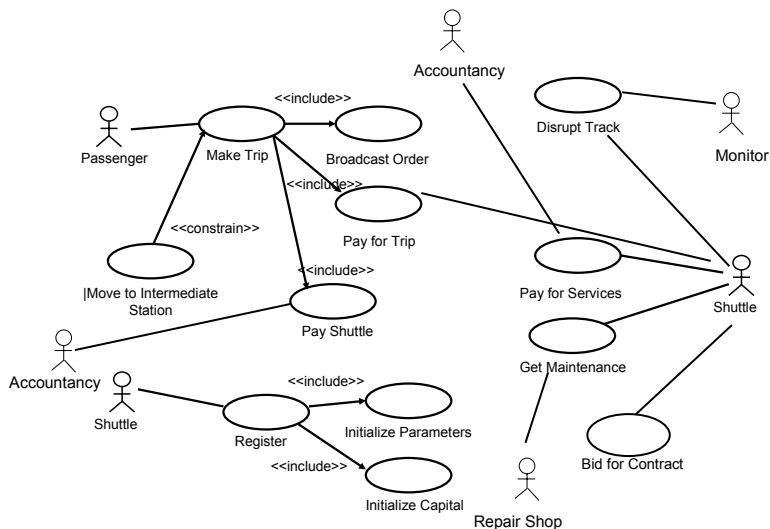
- In a toll gate system, we *must ensure that the vehicle gizmo is read between the moment the vehicle is detected by the system (when it is within ten meters of the toll gate) and enters the toll gate*
- This requirement involves several concerns: gizmo, vehicle, response time, toll gate.



15

Another Example: Shuttle Transports Example with Use Cases

- Miss concerns; lack of composition and impact analysis



16

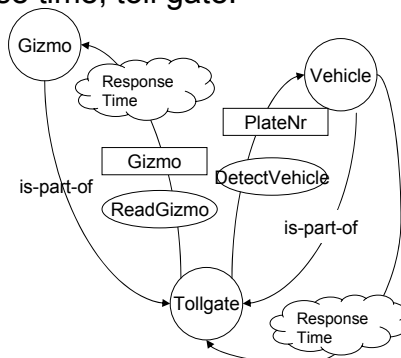
Managing Crosscutting Concerns in Goal-based Models

- Goal-oriented approaches, such as i*, identify several types of concerns, including typical crosscutting concerns
- However:
 - Modular reasoning issues – all concerns are kept intertwined together (e.g., strategic dependency and strategic rational models)
 - Composition is not systematically supported
 - The means-end analysis is done locally to a given agent with the goal of identifying possible operationalizations

17

Motivating Example: Toll Collection System

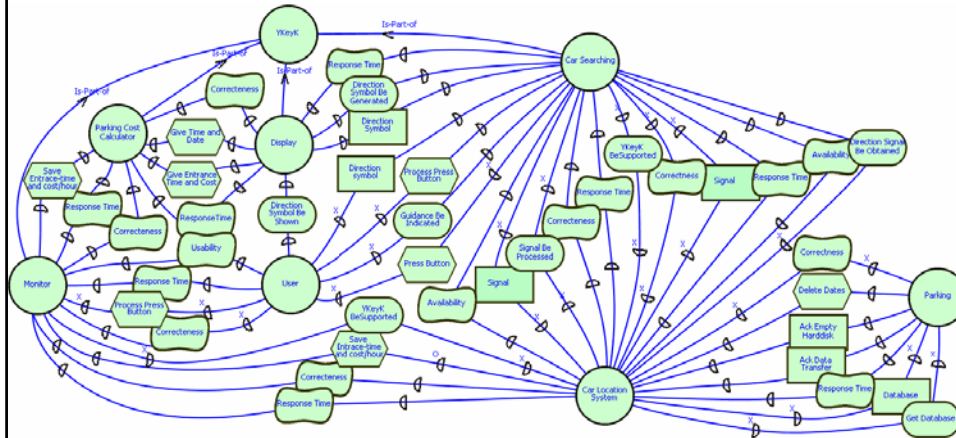
- In a toll gate system, we *must ensure that the vehicle gizmo is read between the moment the vehicle is detected by the system (when it is within ten meters of the toll gate) and enters the toll gate*
- This requirement involves several concerns: gizmo, vehicle, response time, toll gate.



18

Another Example: YKeyK Example with i*

- Lack of modularization and composability mechanisms



19

Problems detected

- Composability
 - Missing systematic mechanism to describe concern influences
- Crosscutting concern identification
 - Missing concerns (use case approach)
- Impact analysis
 - NFR framework and goal models offer support for NFRs
- Modularisation
 - Viewpoints and goals identify crosscutting concerns, but do not cater for compositional context
- Scattering detected
 - The same concern may appear repeatedly in several different viewpoint, use case and goal models hence limiting modular reasoning

20

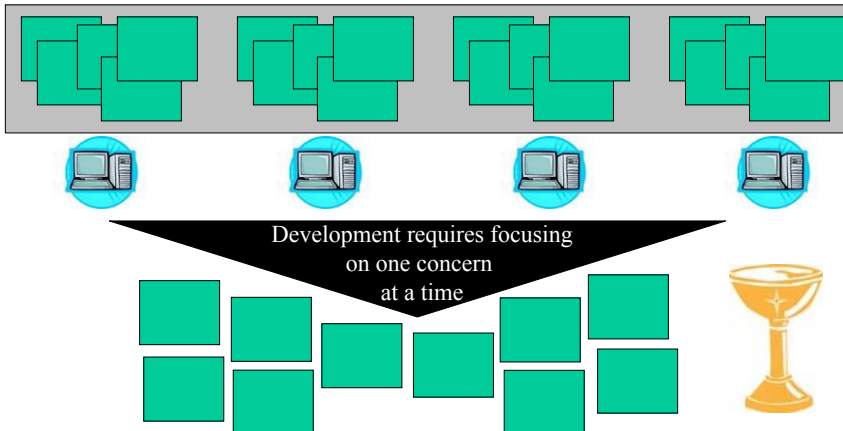
Further Reading

- Discovering Early Aspects, *Baniassad, Clements, Araujo, Moreira, Rashid, Tekinerdogan*, *IEEE Software* 23(1), Jan/Feb 2006
- Domain Models are NOT Aspect Free, *Rashid, Moreira*, *Proceedings of MoDELS/UML Conf. 2006*, Springer.
- Report synthesising state-of-the-art in aspect-oriented requirements engineering, architecture and design, *Chitchyan et al.*, Available from: <http://www.aosd-europe.net>

Introduction to aspect-oriented software development

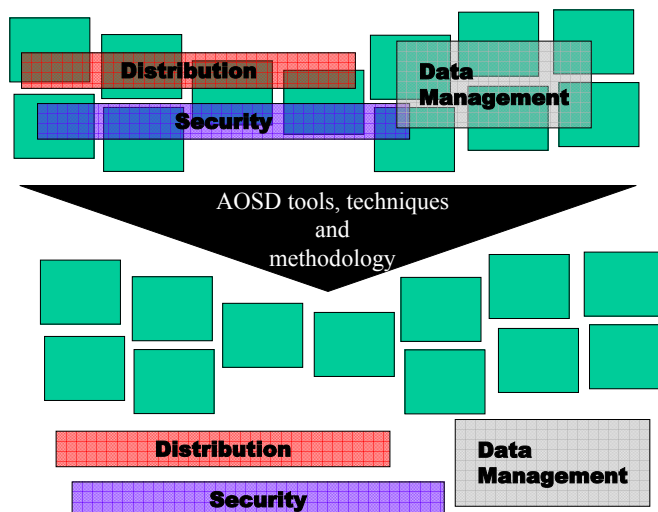
Need for Separation of Concerns

Large ... complex ... distributed software systems



23

Aspect-Oriented Software Development (AOSD)



24

A Definition of AOSD

- **AOSD**: systematic *identification, modularisation, representation* and *composition* of crosscutting concerns [1]

[1] Rashid, A., Moreira, A., Araujo, J. "Modularisation and Composition of Aspectual Requirements", Proceedings of 2nd International Conference on Aspect-Oriented Software Development, ACM, 2003.

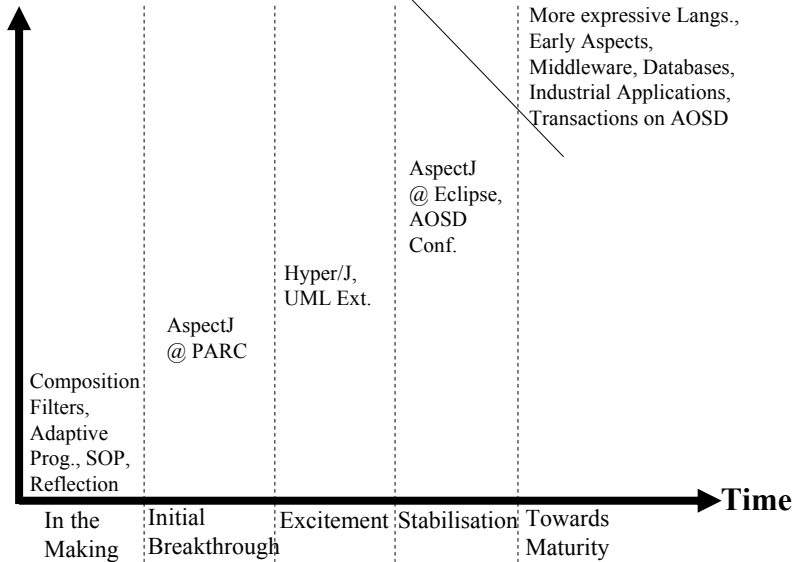
Potential Benefits of AOSD

- Improved ability to reason about problem domain and corresponding solution
- Reduction in application code size, development costs and maintenance time
- Improved code reuse
- Requirements, architecture and design-level reuse
- Improved ability to engineer product lines
- Context-sensitive application adaptation
- Improved modelling methods

H “AOSD is vital to our [IBM Software Group’s] survival”.
 Daniel Sabbah, IBM Vice President, speaking at
 AOSD Conference, Lancaster University, 2004.



Evolution



27

Crosscutting: The Tracing Concern



```
class A {
  // some attributes
  void m1() {
    System.out.println("Entering A.m1( )");
    // method code
    System.out.println("Leaving A.m1( )");
  }

  String m2() {
    System.out.println("Entering A.m2( )");
    // method code
    System.out.println("Leaving A.m2( )");
    // return a string
  }
}
```

```
class B {
  // some attributes
  void m2() {
    System.out.println("Entering B.m2( )");
    // method code
    System.out.println("Leaving B.m2( )");
  }

  int m3() {
    System.out.println("Entering B.m3( )");
    // method code
    System.out.println("Leaving B.m3( )");
    // return an integer
  }
}
```

28

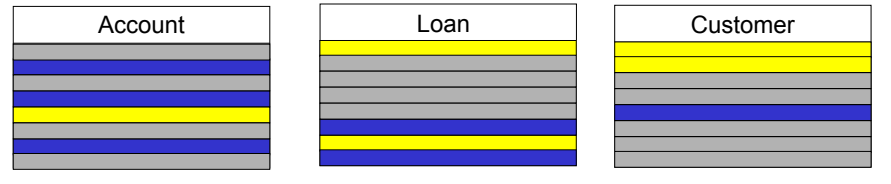
Tangling and Scattering

Primary Functionality

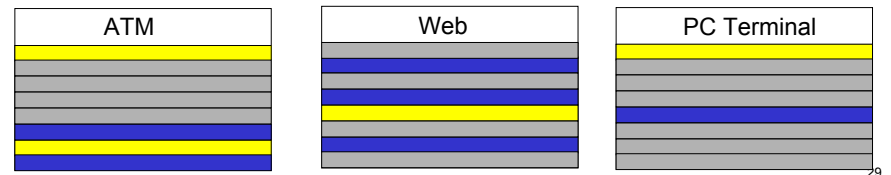
Persistence

Security

Data Classes



User Interface



29

Revisiting the Tracing Example

```
class A {
    // some attributes
    void m1() {
        System.out.println("Entering
        A.m1( )");
        // method code
        System.out.println("Leaving
        A.m1( )");
    }

    String m2() {
        System.out.println("Entering
        A.m2( )");
        // method code
        System.out.println("Leaving
        A.m2( )");
        // return a string
    }
}
```

```
class B {
    // some attributes
    void m2() {
        System.out.println("Entering
        B.m2( )");
        // method code
        System.out.println("Leaving
        B.m2( )");
    }

    int m3() {
        System.out.println("Entering
        B.m3( )");
        // method code
        System.out.println("Leaving
        B.m3( )");
        // return an integer
    }
}
```

30

Wouldn't it be Nice if ...

```
class A {
    // some attributes
    void m1( ) {
        // method code
    }

    String m2( ) {
        // method code
        // return a string
    }
}
```

```
class B {
    // some attributes
    void m2( ) {
        // method code
    }

    int m3( ) {
        // method code
        // return an integer
    }
}
```

aspect Tracing {

when someone **calls** these methods

before the call {System.out.println("Entering " + methodSignature);}

after the call {System.out.println("Leaving " + methodSignature);}

}

31

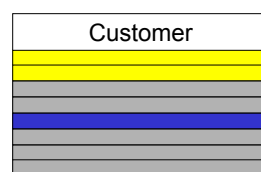
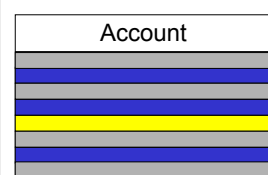
Revisiting the Bank Example

Primary Functionality

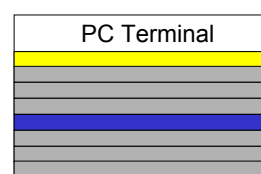
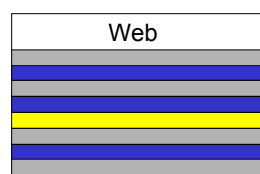
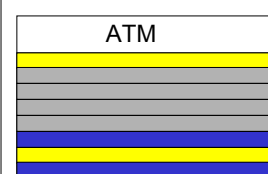
Persistence

Security

Data Classes

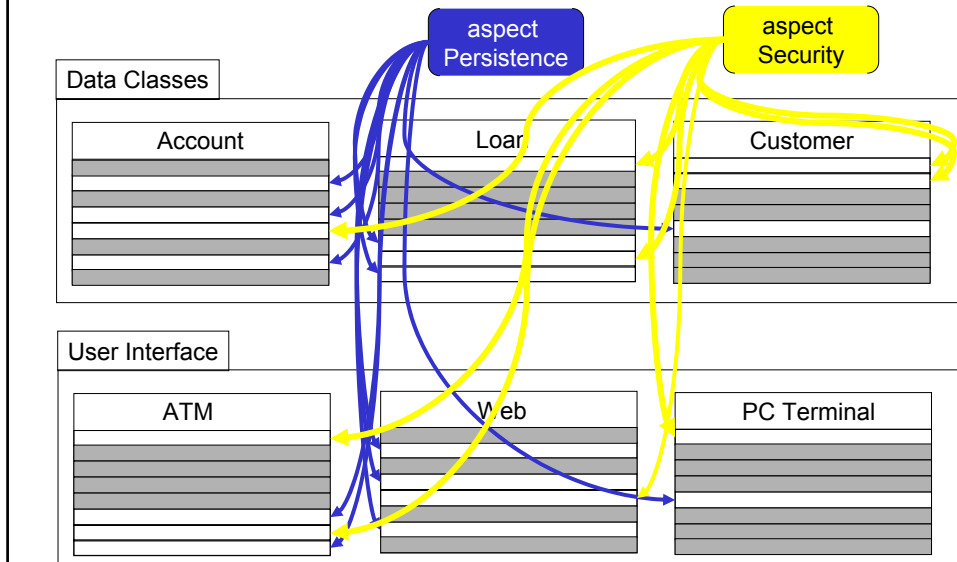


User Interface



32

Wouldn't it be Nice if ...



Crosscutting Concerns Early

- Crosscutting concerns first manifest themselves in requirements – we saw examples of aspects in viewpoint, use case and goal models for requirements
 - Some represent **concerns of the problem domain**
 - Some reflect **stakeholders' real concerns**
- Separation of concerns should begin with stakeholders' concerns

Crosscutting Concerns Later

- Early identification of crosscutting concerns provides:
 - **Broader Vision**: improved localised description of stakeholders' needs, by addressing the crosscutting concerns, as gathered by domain experts
 - **Improved Traceability**: allowing developers to know how aspects arose
 - **Improved Trade-off Handling**: providing insight into the true scope and influence of a concern

35

Main Value of Aspect-Orientation

- **Abstraction**: abstract away from the details of how that crosscutting concern, or *aspect*, might be scattered and tangled with the functionality of other modules in the system
- **Modularization**: keep crosscutting concerns separated regardless of how they affect or influence various other modules in the system, so that we can reason about each module in isolation – **Modular Reasoning**
- **Composition**: the various modules need to relate to each other in a systematic and coherent fashion so that one may reason about the global or emergent properties of the system – **Compositional Reasoning**

36

What is Aspect-Oriented Requirements Engineering About?



- Bringing aspect-orientation benefits of
 - Abstraction
 - Modularity
 - Composabilityto improve modular and compositional reasoning about stakeholders' requirements
- To address the scattering, tangling and composability issues during requirements engineering (that we highlighted early on)

37

Further Reading



- AOSD Wiki at: <http://www.aosd.net>
- Introduction to AOSD White Paper and AOSD Ontology available at: <http://www.aosd-europe.net>
- Communications of the ACM, Special Section on AOP, 44(10), October 2001
- IEEE Software, Special Section on AOP, 23(1), Jan/Feb 2006
- Aspect-Oriented Software Development, *Filman, Elrad, Clarke, Aksit (eds), Addison-Wesley 2004*

38

Overview of aspect-oriented requirements engineering

Aspect-Oriented Requirements Engineering

- Improved support for separation of crosscutting functional and non-functional properties during requirements engineering
 - A better means to identify and manage conflicts arising due to tangled representations
- Identify the mapping and influence of requirements-level aspects on artefacts at later development stages
 - Establish critical trade-offs even before the architecture is derived
- Trace aspectual requirements and their trade-offs to architecture and subsequently all the way to implementation

Improved understanding of the problem and ability to reason about it

Managing Crosscutting Concerns in Requirements (1)

- To find crosscutting in requirements, look for behavioural terms or concepts that are mentioned in more than one location
- 1. *Pay interest* of a certain percent on each account making sure that the *transaction is fully completed* and an *audit history is kept*.
- 2. Allow customers to *withdraw* from their accounts, making sure that the *transaction is fully completed* and an *audit history* is kept.

41

Managing Crosscutting Concerns in Requirements (2)

- Separate each of those concepts into a section of their own

1. *Pay interest* of a certain percent on each account

2. Allow customers to *withdraw* from their accounts

3. To *fully complete a transaction*...

4. To *maintain an audit history*...

42

Managing Crosscutting Concerns in Requirements (3)

- Composition specifies the crosscutting relationship, showing how crosscutting concerns affect base concerns

1. *Pay interest of a certain percent on each account*

Fully complete pay interest and withdraw

3. *To fully complete a transaction...*

2. *Allow customers to withdraw from their accounts*

Audit pay interest and withdraw

4. *To maintain an audit history...*

43

Identifying Aspects

- A challenging problem
- Requirements can be obtained from different sources
 - Interviews, manuals, Sketches
 - Standardisation documents
 - Organisational procedures
- And can be represented in a variety of forms
 - Natural Language
 - Diagrammatic models
 - Formal specifications
- Think multi-disciplinary when developing techniques for aspect identification
 - Natural language processing
 - Ethnography techniques

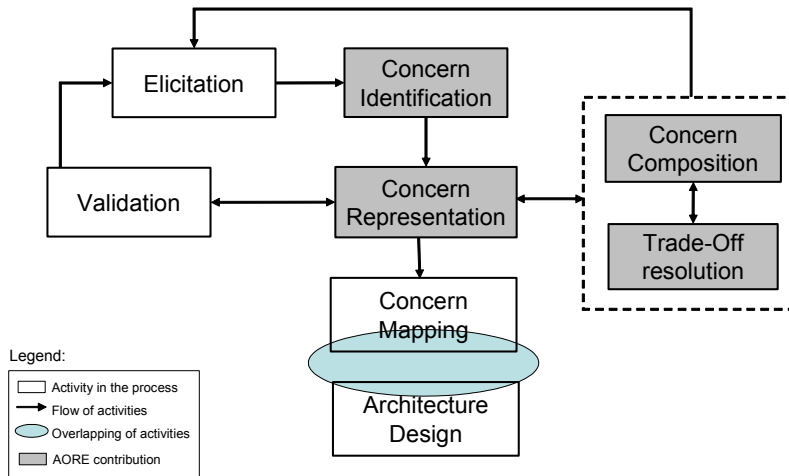
44

Weaving/Composition

- What does it mean to weave or compose aspectual requirements?
- *Projecting* the constraints and influences of aspectual requirements on other system requirements
- *A synthesis of the various projections* that provides a deeper understanding of the system that we want to develop
 - Critical needs of the stakeholders
 - Identification of the key properties of a system
 - Various trade-offs affecting the system

Mapping and Traceability

- How do Early Aspects map onto later development stages?
- Is an Early Aspect always a design and implementation aspect?
- How can we ensure forward and backward traceability between early aspects and corresponding design and implementation?
- How do we know that the early aspect trade-offs are preserved and respected by the design and implementation?



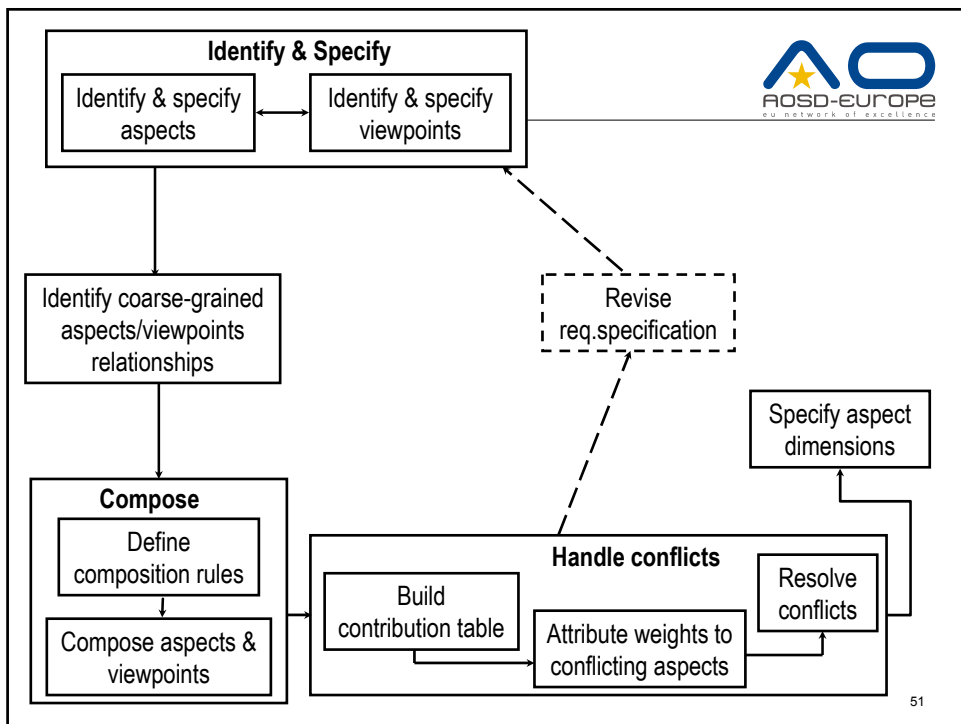
Further Reading

- Early Aspects Portal: <http://www.early-aspects.net>
- Discovering Early Aspects, *Baniassad, Clements, Araujo, Moreira, Rashid, Tekinerdogan, IEEE Software 23(1), Jan/Feb 2006*
- Special Issue on Early Aspects, *IEE proceedings - Software Engineering - Volume 151, Issue 04, August 2004, (Rashid, Moreira, Tekinerdogan (eds))*
- Report synthesising state-of-the-art in aspect-oriented requirements engineering, architecture and design, *Chitchyan et al., Available from: <http://www.aosd-europe.net>*

AORE for viewpoints-based requirements engineering

A Viewpoint-based Model for AORE

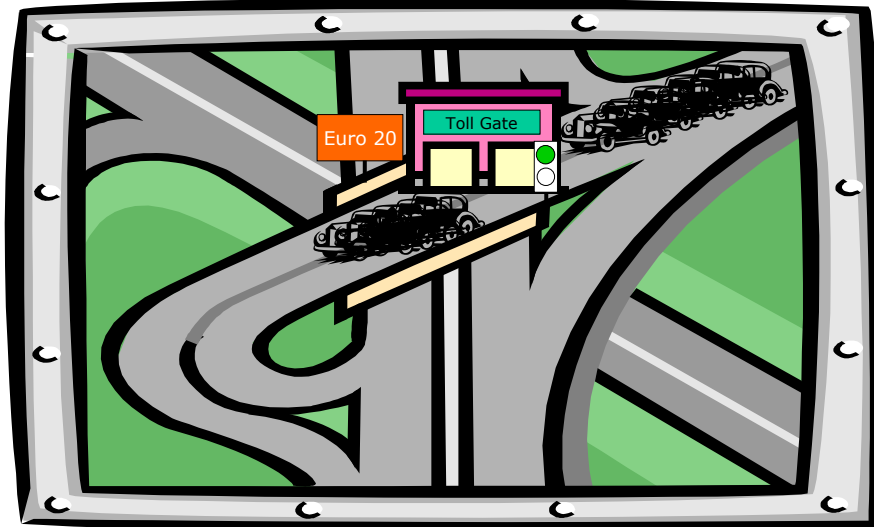
- Viewpoints specify stakeholder requirements
- Aspects crosscut the viewpoints
- Composition information separated from aspects
- Fine-grained composition relationships
 - Observe the influence of an aspectual requirement on individual viewpoint requirements
- Explicit support for establishing aspectual trade-offs and subsequent negotiations



Requirements Representation and Tool Support

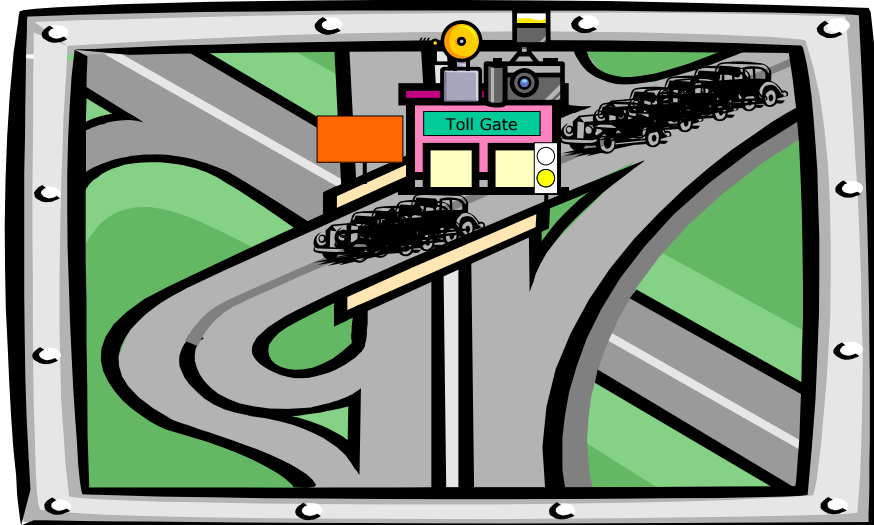
- XML
 - Extensible language for specification of viewpoints, aspects and their composition
- EA-Miner: NLP-based tool for identifying aspects in requirements
- ARCaDe: Aspectual Requirements Composition and Decision Support Tool
 - DOM, SAX and eXist Native XML database

Toll Collection System: Authorised Vehicle

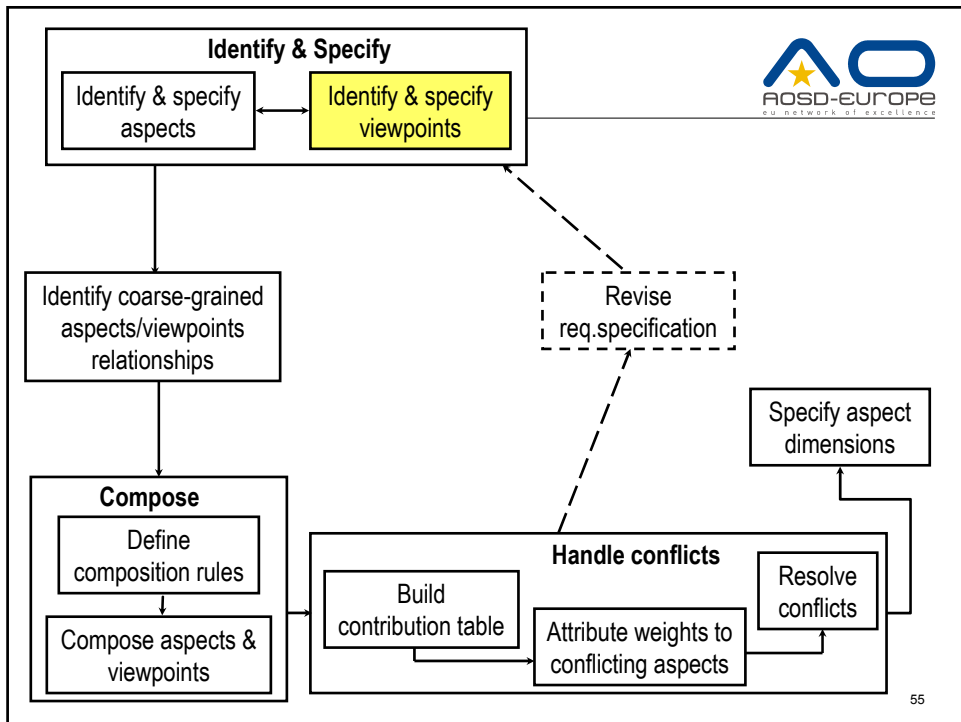


53

Toll Collection System: Unauthorised Vehicle



54



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

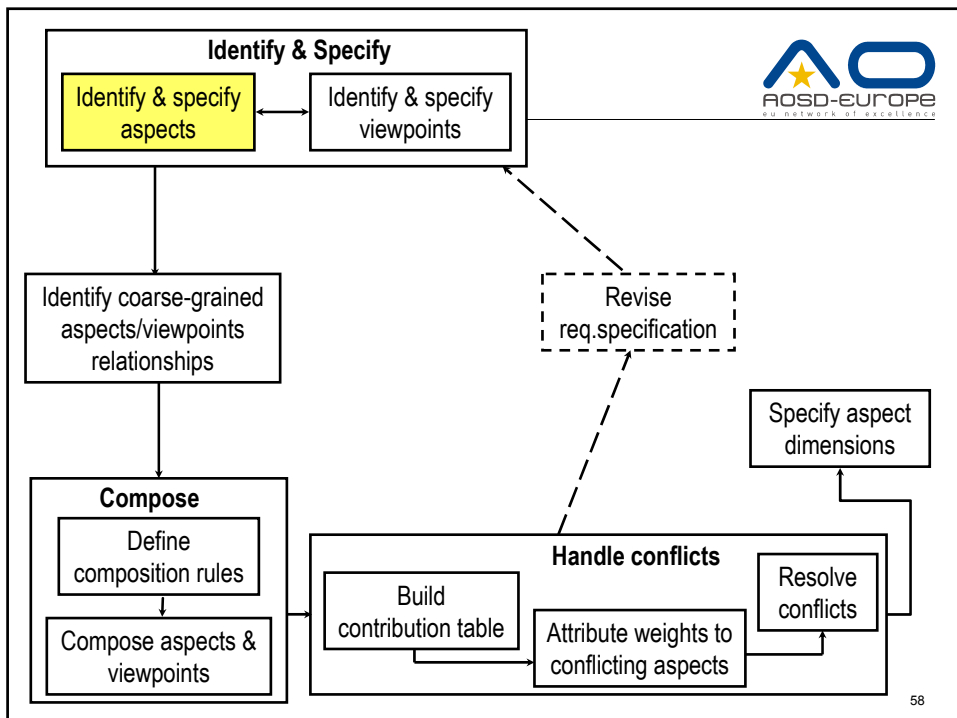
Vehicle Viewpoint

```

<?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

57



58

Aspects

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

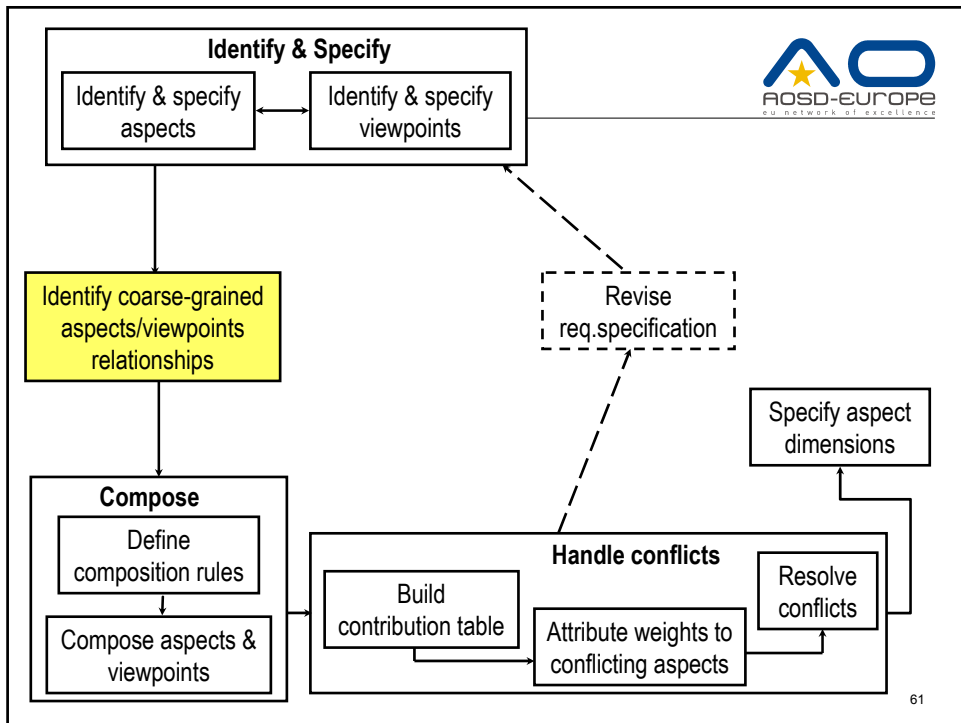
59

Response Time Concern

```
<?xml version="1.0" ?>
- <aspect 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>
</aspect >
```

Subrequirements

60



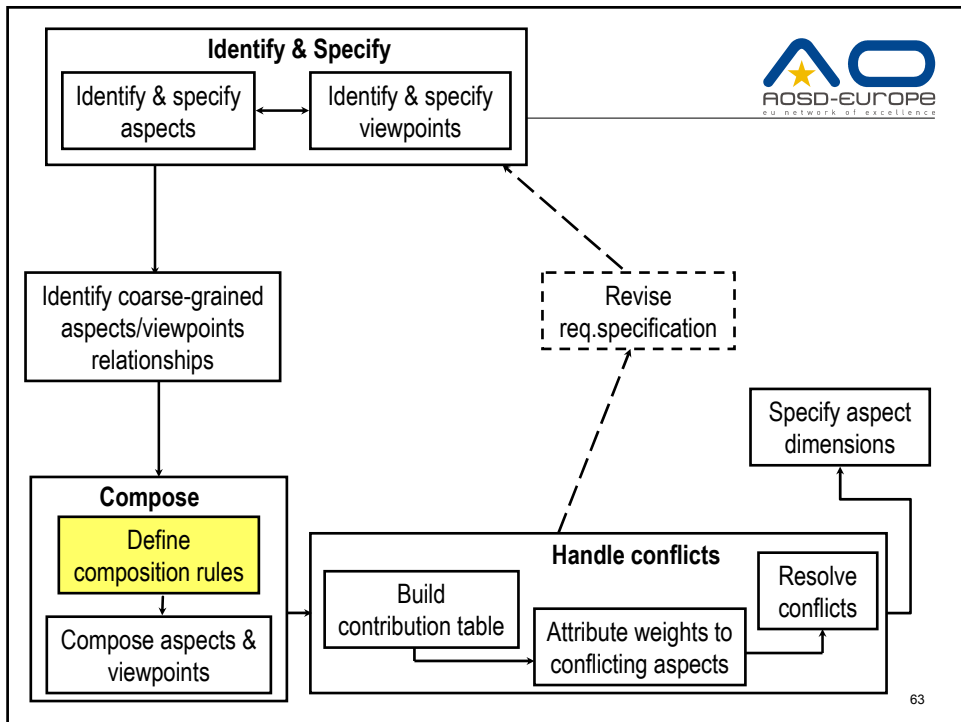
Aspect/Viewpoint Relationships



VP Aspects	Pol	Gz	DS	ATM	TG	...	Vh	UV	VO	...	Adm
Response Time		x		x	x		x	x			
Availability		x		x	x						x
Security	x		x	x					x	x	x
Legal Issues	x									x	
Compatibility	x		x	x							
Correctness	x	x	x		x				x	x	
Multi Access		x		x	x		x	x			x

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

62



Composition Rule for Response Time Requirements



```
<?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>

```
...
```

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

Subrequirements must be explicitly excluded or included

Describes whether another (or a set of) viewpoint requirement must be *satisfied* or merely the *constraint* specified *fulfilled*

The system needs to react in-time to read the gizmo identifier

- “**Response Time** requirement **1.1** must be **enforced** **between** requirements **Vehicle 1** and **Vehicle 2** with the outcome that **Gizmo** requirement **1** including its children is **satisfied**”

The gizmo identifier is read by the system

1. The vehicle enters the system when it is within ten meters of the toll gate
2. The vehicle enters the toll

Composition Rule for Response Time Requirements

```

- <Requirement aspect="ResponseTime" id="1.2">
  - <Constraint action="enforce" operator="between">
    <Requirement viewpoint="Gizmo" id="1" children="include" />
    <Requirement viewpoint="Vehicle" id="3" />
  </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 <i>all</i> is specified.	ANY

Composition Semantics: Constraint Operators

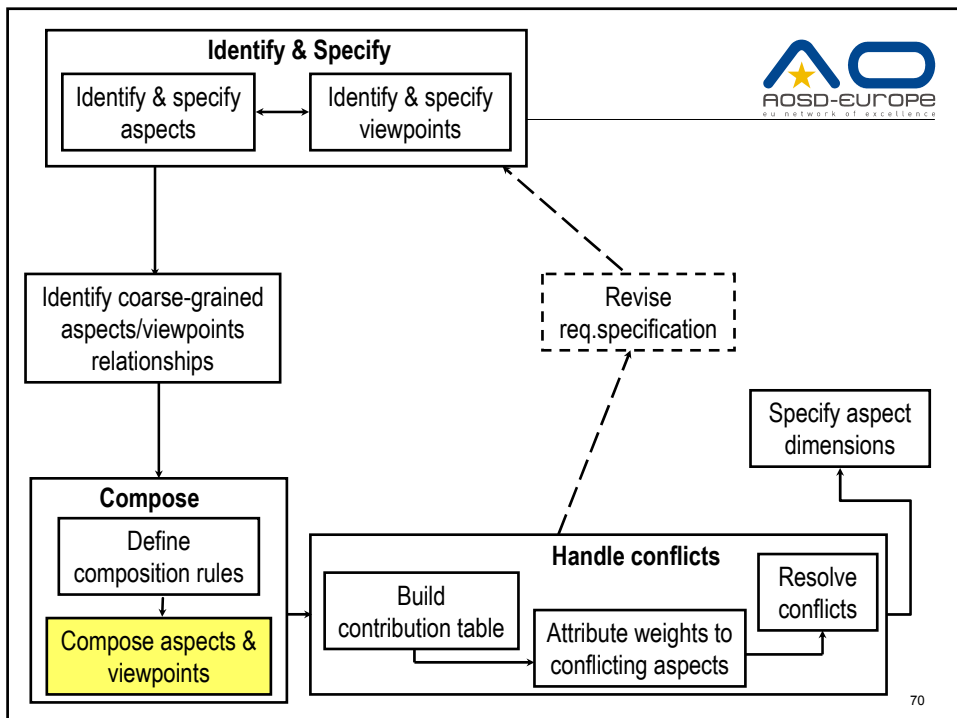


Constraint Operator		Action	Valid aspect: action-operator combinations
Type	Description		
during	Describes the temporal interval during which a set of reqs. is being satisfied.	ensure	Availability: ensure-during
between	Describes the temporal interval falling between the satisfaction of two reqs.	enforce	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 Security: provide-for 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

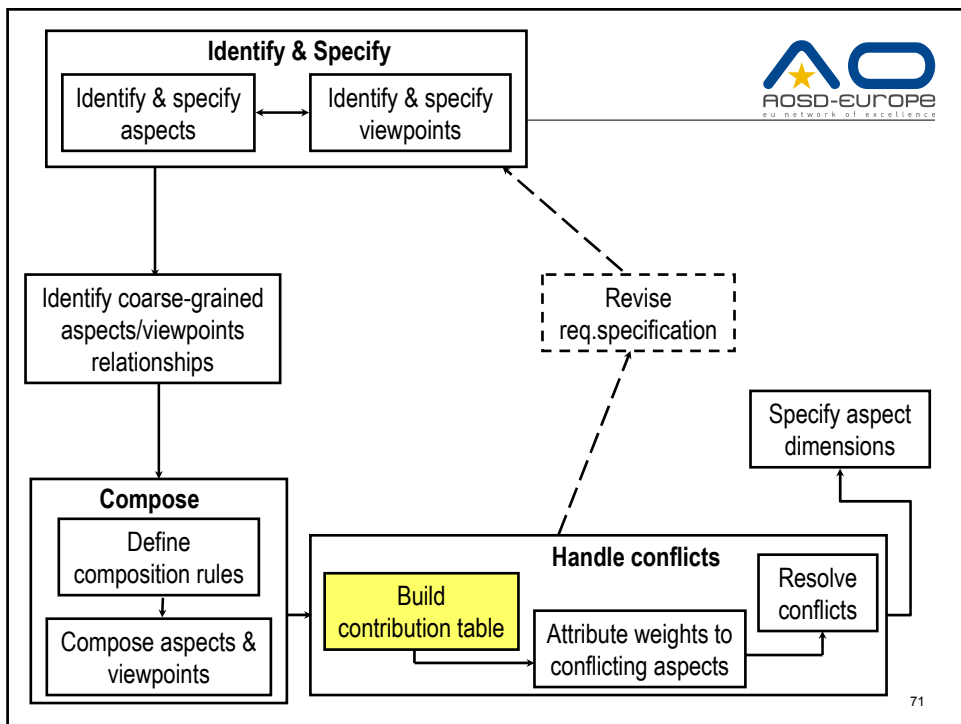
Composition Semantics: Outcome Actions

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

69



70



Trade-off Analysis (1)

- Basic questions
 - Are the solutions that are being suggested as good as possible?
 - How much must I give up to get a little more of what I want most?
- How can we define a good / better outcome?
 - Reduced time of response
 - Improved security access
 - Increase number of clients being served simultaneously

Trade-off Analysis (2)

- Finding the combination that better satisfies the stakeholder goals
- What can actually be done given reality and resource constraints?
 - Use sophisticated techniques such as MCDM, fuzzy logic, etc.
 - Use some simple way to achieve a ranking between conflicting concerns (with the stakeholders help)

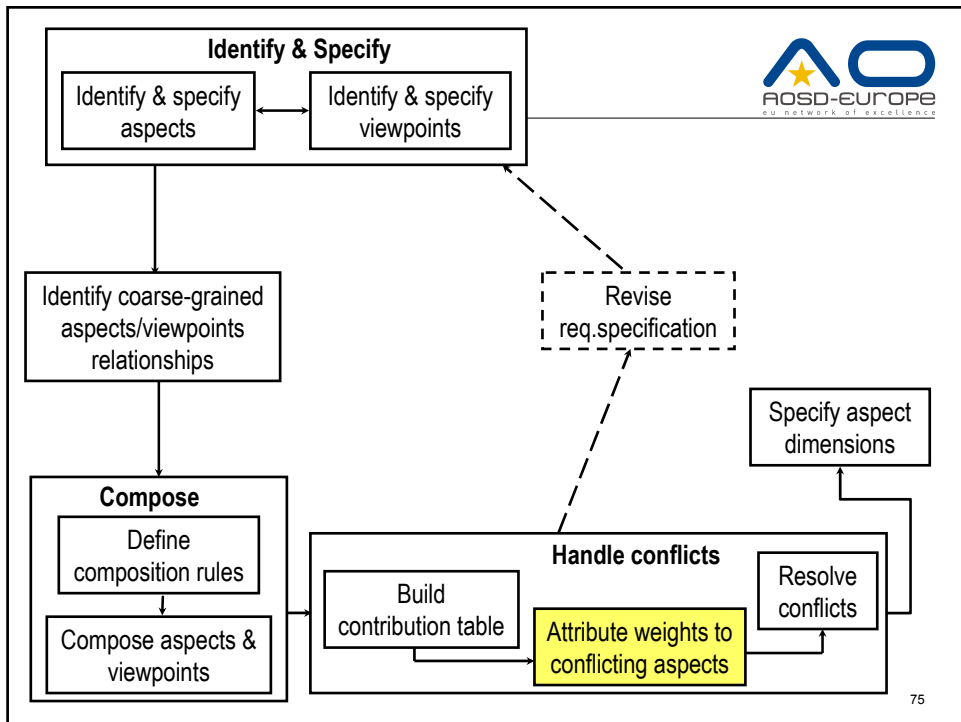
73

Identify contributions between aspects

- Building a contribution matrix where each aspect may contribute negatively (-) or positively (+) to the others (empty cells represent "don't care" contributions)

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

74



Attribute Weights to Conflicting Aspects (1)

- Attributing weights to those aspects that contribute negatively to each other
 - Priority of an aspect in relation to a set of stakeholders' requirements
 - Extent to which an aspect may constrain another concern

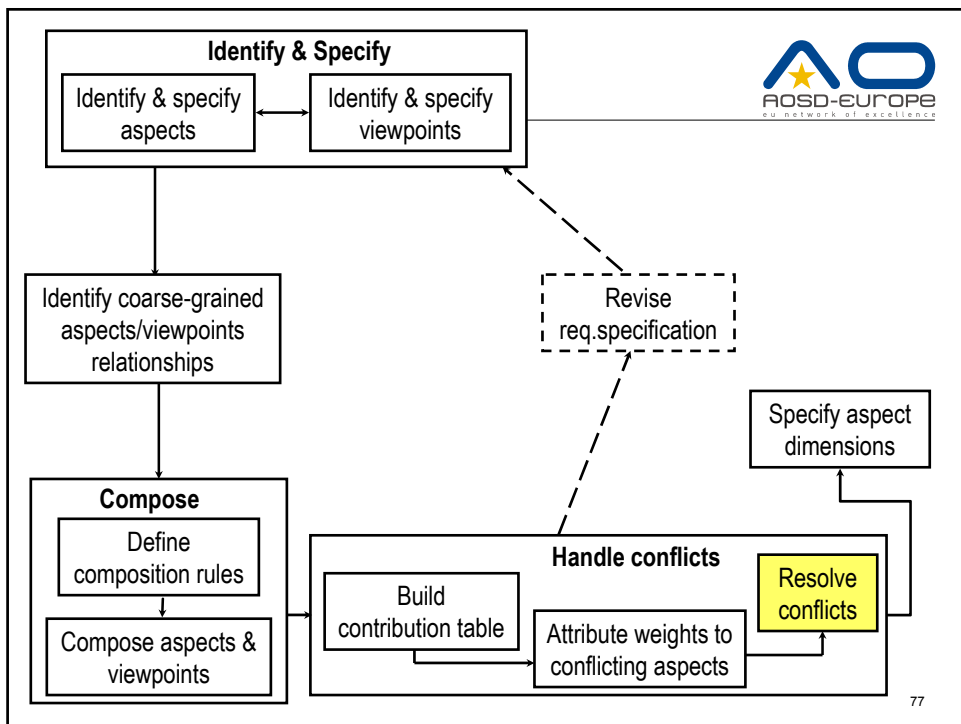
Very important takes values in the interval $]0,8 \dots 1,0]$

Important takes values in the interval $]0,5 \dots 0,8]$

Average takes values in the interval $]0,3 \dots 0,5]$

Not so important takes values in the interval $]0,1 \dots 0,3]$

Do not care much takes values in the interval $[0 \dots 0,1]$

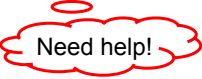


Conflict Resolution


AOSD-EUROPE
eu network of excellence

React in time vs display correct amount

Viewpoints Aspects	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		x		x	x	x	x	x	x	
Security	x		x	1.0						
Legal Issues	x							x		
Compatibility	x		x	x						
Correctness	x	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



Need help!



Easy!

78

Summary

- Viewpoint-oriented approaches are enhanced with aspect-orientation by:
 - Providing an improved separation of concerns – viewpoints and aspects are kept in separate modules hence facilitating **modular reasoning**
 - Identifying crosscutting relationships among concerns and representing them via composition rules
 - Offering the possibility to compose concerns and project their mutual dependencies and interactions hence facilitating **compositional reasoning**
 - Offering means for an early trade-off analysis
 - Identifying conflicts and realising trade-offs much earlier

79

Further Reading

- Modularisation and Composition of Aspectual Requirements, *Araujo, Rashid, Moreira, Proc. AOSD Conference 2003, ACM.*
- EA-Miner: A Tool for Automating Aspect-Oriented Requirements Identification, *Sampaio, Chitchyan, Rayson, Rashid, Proc. Automated Software Engineering Conf. 2005, ACM.*
- Multi-dimensional Separation of Concerns in Requirements Engineering, *Moreira, Rashid, Araujo, Proc. Requirements Engineering Conf., 2005, IEEE CS Press.*

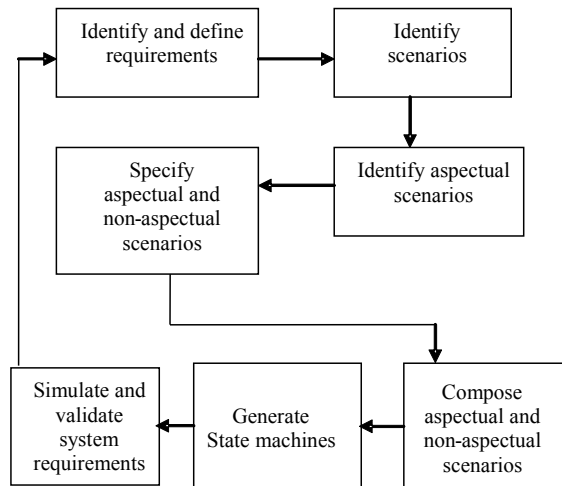
80

AORE for scenario-based requirements engineering

A Scenario-based Approach for AORE

- Scenarios are commonly used in RE because they are easily understood by stakeholders.
- A scenario is an example trace of desired or existing system behavior.
- A complete and consistent set of scenarios can be difficult to specify (e.g., exception, failure cases).
- Many of these are aspectual in the sense that they crosscut other scenarios.
- Our process provides a way to describe aspectual and non-aspectual scenarios independently and then to merge them together for the purpose of validating the complete set of scenarios.

The Scenario-based Approach

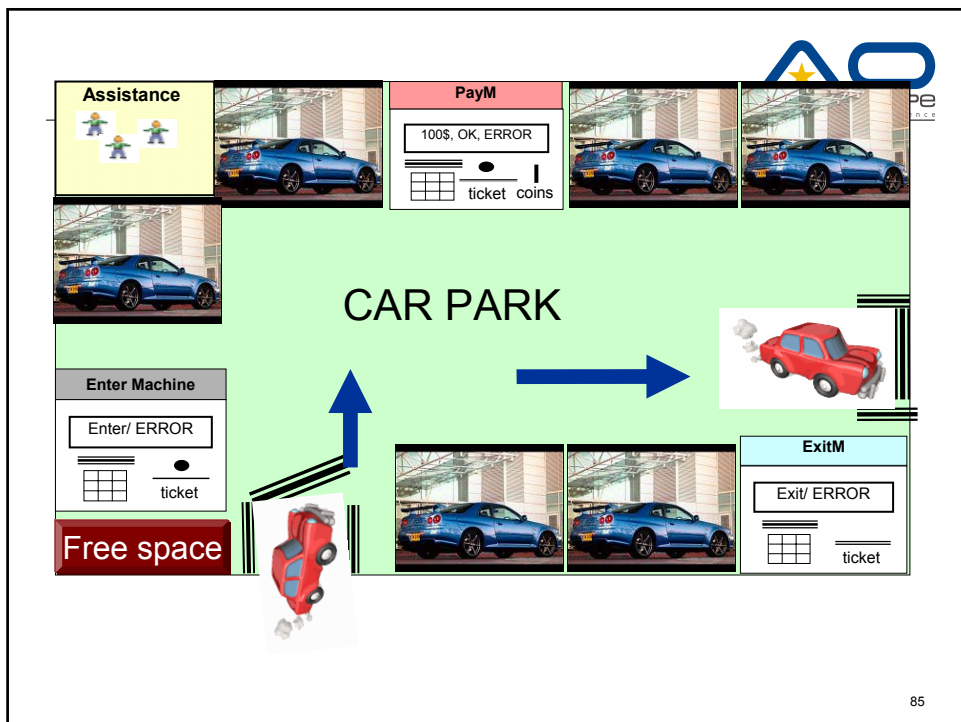


83

The Car Parking System

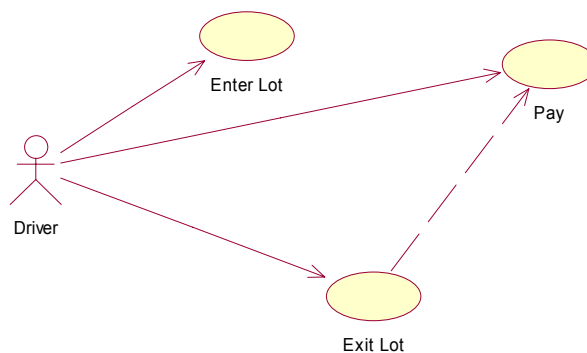
- A client has to get a ticket from a machine after pressing a button
- Afterwards, the car is allowed to enter and park in an available place
- The system has to control if the car parking is full or if it still has places left
- To leave the parking place, s/he has to pay the ticket obtained in a paying machine
- The amount depends on the time spent. After paying the client can leave by inserting the ticket in a machine which will open the gate
- Regular users of the parking system may pre-purchase time and enter/exit by inserting a card and PIN number which will result in money being deducted automatically from the user's account

84



85

Use Case Diagram for the Car Parking System



86

Non-Aspectual Scenarios

I1	Enter Lot, parking lot has space
I2	Enter Lot, parking lot has no space
I3	Enter Lot, regular user types in PIN and enters
I4	Exit Lot, driver inserts ticket; ticket paid
I5	Exit Lot, driver inserts ticket; ticket not paid
I6	Exit Lot, driver has no ticket
I7	Exit Lot, grace period from paying ticket exceeded
I8	Exit Lot, regular user types in PIN and exits
I9	Exit Lot, driver types in PIN but insufficient funds in account
I10	Pay, driver inserts ticket and correct money
I11	Pay, driver adds money to PIN card

87

Aspectual Scenarios

A1	Machine is broken
A2	Ticket cannot be read
A3	PIN incorrect

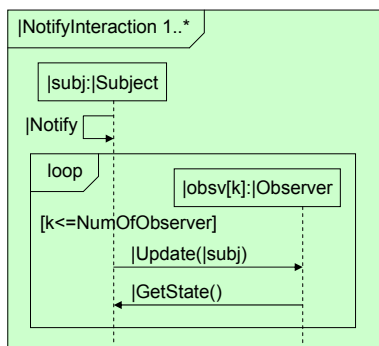
88

Pattern Specifications

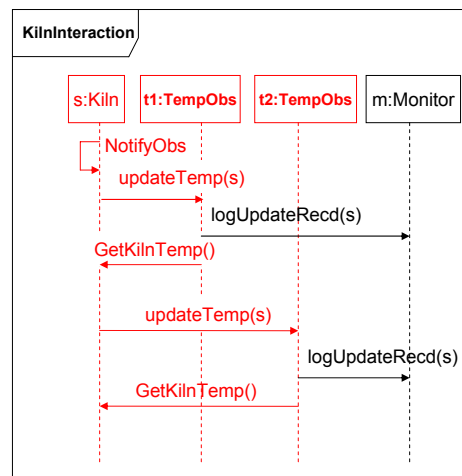
- A Pattern Specification (PS) describes a pattern of structure or behavior and is defined in terms of roles
- A role specifies properties that a UML model element must have if it is to be part of the UML model
- A PS can be instantiated by assigning UML model elements to the roles in the PS
- A model conforms to a PS if its model elements that play the roles of the PS satisfy the properties defined by the roles

89

Conforming Sequence Diagram



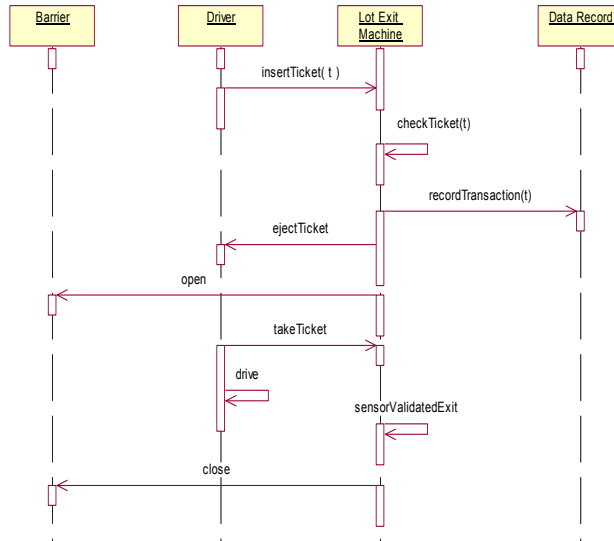
IPS



Conforming SD

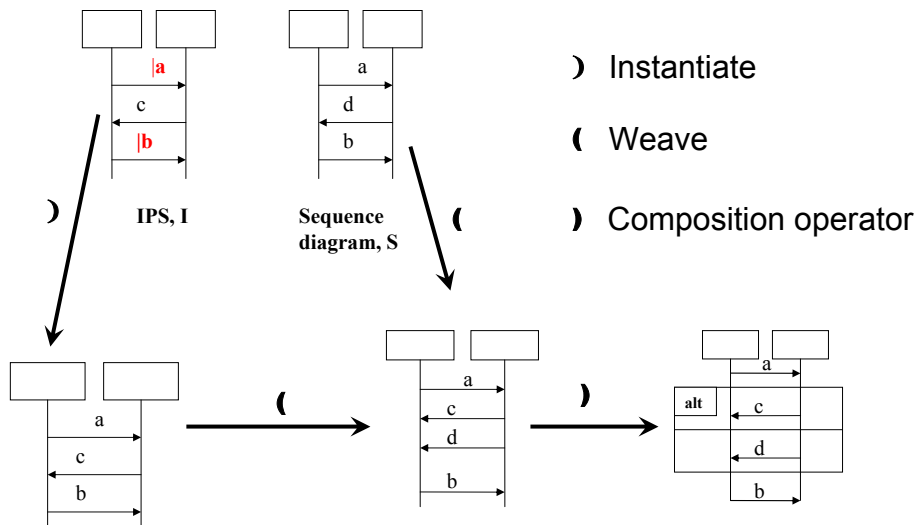
90

Sequence Diagram for Exiting with Paid Ticket



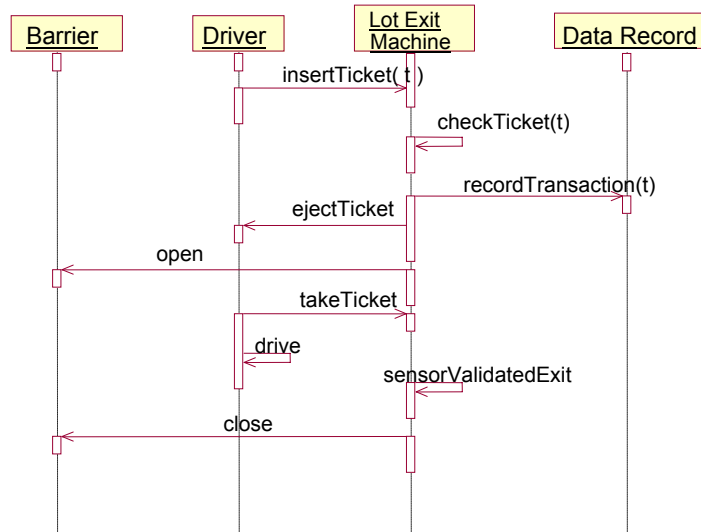
91

Composition at Interaction Level



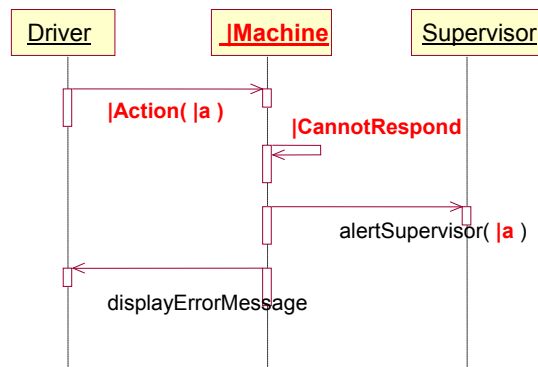
92

Car Parking Example



93

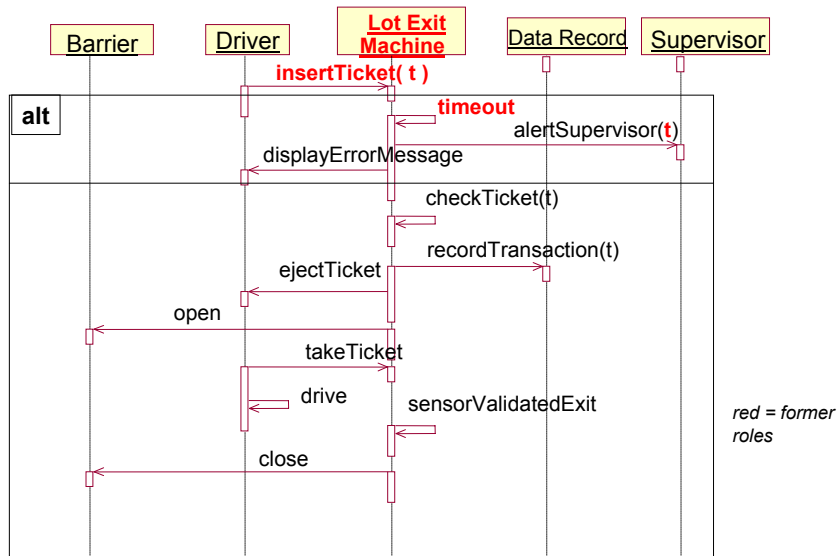
Instantiation



|Machine binds to Lot Exit Machine
 |Action binds to insertTicket
 |a binds to t
 |CannotRespond binds to timeout

94

Composed Interaction



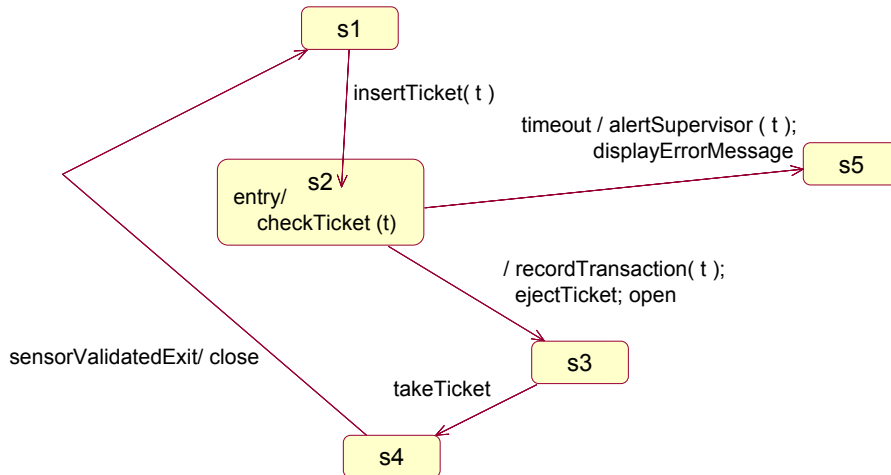
95

Composition Operators

- **OR**: alternative interactions with choice point to decide
- **AND**: interactions occur concurrently
- **IN**: insert one interaction inside the other
- Others: further work...

96

State Machine Generated



97

Further Reading

- Modeling and Composing Scenario-based Requirements with Aspects, *Araujo, Whittle, Kim, Proc. Requirements Engineering Conf. 2004, IEEE CS Press.*
- Scenario Modeling with Aspects, *Whittle, Araujo, IEE Proceedings – Software 151(4), 2004 (Special Issue on Early Aspects)*

98

From aspectual requirements to architecture

Current Software Architectures

- Current software architecture design methods **do not distinguish** between conventional architectural concerns and **architectural concerns that crosscut multiple architectural components**
- The **risk is** that potential aspects might be easily overlooked during the software architecture design
 - leading to poorly designed / unstable architecture
 - also the overlooked aspects might remain unsolved at the design and programming level
 - leading to tangled code, diminishing quality factors that the architecture analysis methods attempt to verify

Issues on Mapping and Traceability

- Is a Requirements Aspect always a design and implementation aspect?
- How do Requirements Aspects map onto later development stages?
- How can we ensure forward and backward traceability between requirements aspects and corresponding architecture, design and implementation?
- How do we know that the requirements aspect trade-offs are preserved and respected by the design and implementation?

101

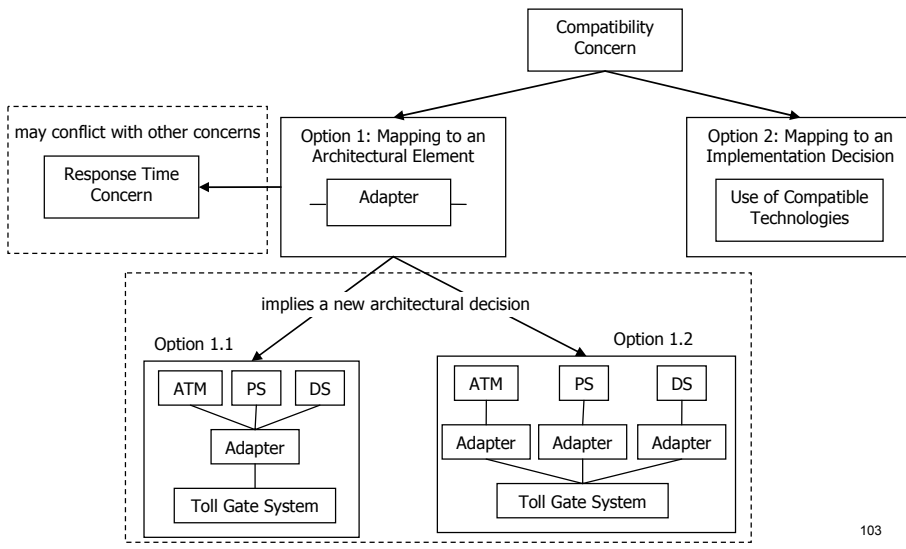
Is a Requirements Aspect always a design and implementation aspect?

```
<aspect name="Compatibility">
  <Requirement id="1"> The system must be compatible with systems used to:
    <Requirement id="1.1">activate and reactivate gizmos <Requirement>
    <Requirement id="1.2">deal with infraction incidents <Requirement>
    <Requirement id="1.3">charge for usage;<Requirement>
  </Requirement>
</aspect>

<Composition>
  <Requirement aspect="Compatibility" id="1.1">
    <Constraint action="ensure" operator="with">
      <Requirement viewpoint="ATM" id="all" />
    </Constraint>
    <Outcome action="fulfilled" />
  </Requirement>
  <Requirement aspect="Compatibility" id="1.2">
    <Constraint action="ensure" operator="with">
      <Requirement viewpoint="Police" id="all" />
    </Constraint>
    <Outcome action="fulfilled" />
  </Requirement>
  ...
</Composition>
```

102

Mapping Compatibility Concern to Architecture



103

How do Requirements Aspects Map onto Later Development Stages?

- A crosscutting in requirements may be resolved as non-software solution (e.g. for some business *hiring a security guard to satisfy a Security concern* → no mapping to software architecture)
- Architectural Decision (e.g., the architectural topology for Compatibility concern)
- Architectural Module or an element of a module which has localised influence in architecture (e.g., a method checking correctness or some processing within a component)
- Aspectual Architectural Modules which influence a number of other architectural components (e.g., a module providing encryption and decryption functionality)
- **NOTE: not all architecture/ design/ implementation aspects come from requirements!**

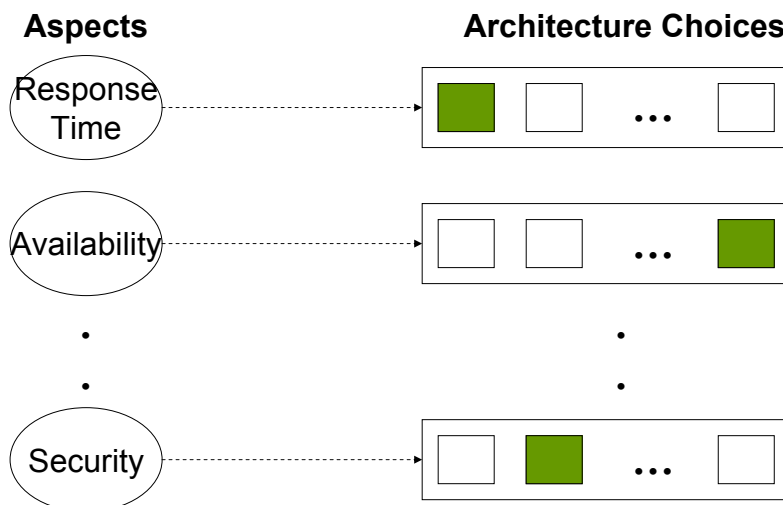
104

Aspect Traceability

- As always need Tool Support for Traceability, but
 - *Dedicated modules* for crosscutting concerns at various levels help with direct mapping where appropriate
 - *Composition specifications* at various levels help in tracing influences:
 - Examples of composition specifications in Architecture are bindings in ADL supported in connectors or dedicated composition languages as part of traceability templates
 - Correspondence between meta-models of requirements and architecture (etc.) elements helps with mapping and traceability guidelines.

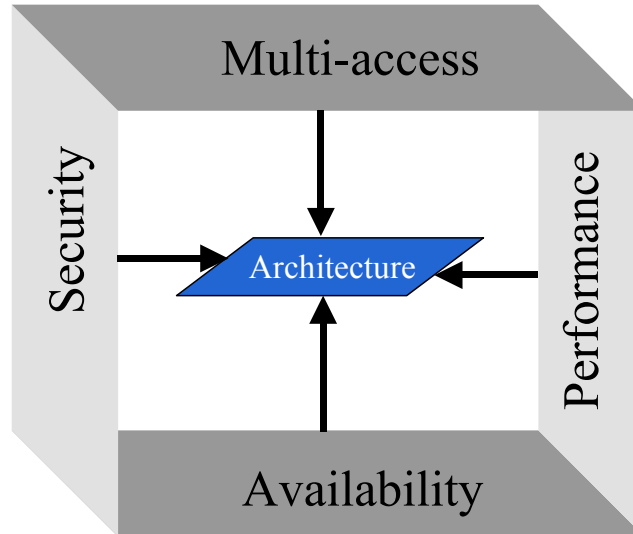
105

Aspects Driving Architectural Choices



106

Where Does the Final Architecture Reside?



107

Are Trade-offs Preserved and Respected?

- Need to connect aspectual requirements to their concrete realisations in design and implementation
 - Need to know that trade-offs established at requirements level are preserved in the implementation
- Needs precise semantics for requirements
- Proof obligations may be used for:
 - Formal verification of the resultant system (e.g., via model checking)
 - Deriving test cases

108

Aspect Proof Obligations

- Temporal logic formulas about aspects
- Group individual formulas to those treating each aspect
- Must treat potential conflicts among aspects (as identified in requirements)
- Show original specification of system without aspects is not violated

Temporal Logic Formulas

- Gp : from now on, p is true
- Fp : eventually, p will be true
- pUq : p is true until q becomes true
- Can be input for formal methods tools
- $G((veh-ent-sys \rightarrow ((Fveh-ent-toll) \wedge ((\neg veh-ent-toll) U read-gizmo))) \wedge (read-gizmo \rightarrow gizmo-effects) \wedge (time(veh-ent-toll) - time(veh-ent-sys) < N))$

Conflict Analysis

- One viewpoint may be constrained by multiple aspects with contradictory requirements
 - Availability (→ backup system) vs Response Time
- Often requires preferring one over the other or weakening for real conflicts
- Solution: weakened properties for conflict resolution
 - Weak Resp. Time: ... $\text{time}(E2) - \text{time}(E1) < N + \text{Extra}$

111

The Resulting Proof Obligations

- Collection of temporal formulas from the requirements with predicates instantiated
- Possibly weakened versions for conflicts
- Assertion of non-interference among aspects
- Assertion that desirable properties of the original system are still true in augmented one (aspects “do no harm”)

112

Further Reading

- Multi-dimensional Separation of Concerns in Requirements Engineering, *Moreira, Rashid, Araujo, Proc. Requirements Engineering Conf., 2005, IEEE CS Press.*
- Discovering Early Aspects, *Baniassad, Clements, Araujo, Moreira, Rashid, Tekinerdogan, IEEE Software 23(1), Jan/Feb 2006*
- Driving and Managing Architectural Decisions with Aspects, *Garcia, Batista, Rashid, SantAnna, Proc. Workshop on SHARing and Reusing architectural Knowledge (SHARK'2006), ACM.*
- From Aspectual Requirements to Proof Obligations for Aspect-Oriented Systems, *Katz, Rashid, Proc. Requirements Engineering Conf., 2004, IEEE CS Press.*
- Relating AO Requirements to AO Architectures, *Chitchyan, Pinto, Fuentes, Rashid, Workshop on Early Aspects @ OOPSLA 2005.*

113

Concluding Remarks

- Aspect-oriented requirements engineering **IS** requirements engineering
- Don't throw away your existing RE practices
 - AORE aims at complementing these to improve modular and compositional reasoning about stakeholders' needs
- Improved understanding of the problem domain helps design better solutions
 - Clearer mapping of otherwise scattered and tangled requirements to architecture and beyond
 - Early trade-off analysis and resolution
 - Improved traceability of stakeholders' concerns to the solution and its implementation

114