Aspect-Oriented Requirements Engineering Explained

Prepared by Yuri Chernak, Ph.D.
Valley Forge Consulting, Inc
Email: ychernak@yahoo.com

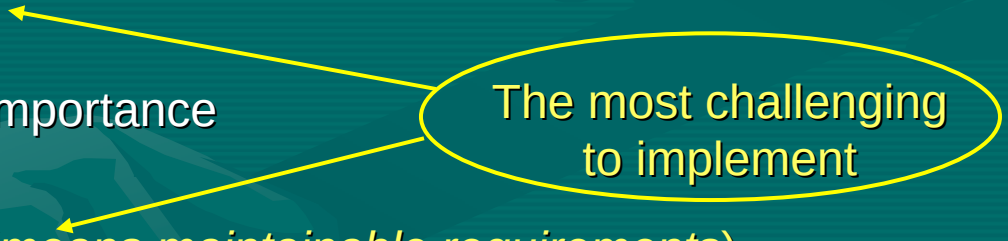
Presentation Outline

- Introduction
- Part 1. AORE Analysis Techniques
- Part 2. AORE Specification Techniques
- Part 3. Using RCT for Change Impact Analysis
- Part 4. Using RCT for Test Coverage Assessment
- Presentation Summary
- Appendix A: Descriptions of Common Crosscutting Concerns
- Appendix B: Composition Modeling with UML

Introduction

Characteristics of Good Software Requirements

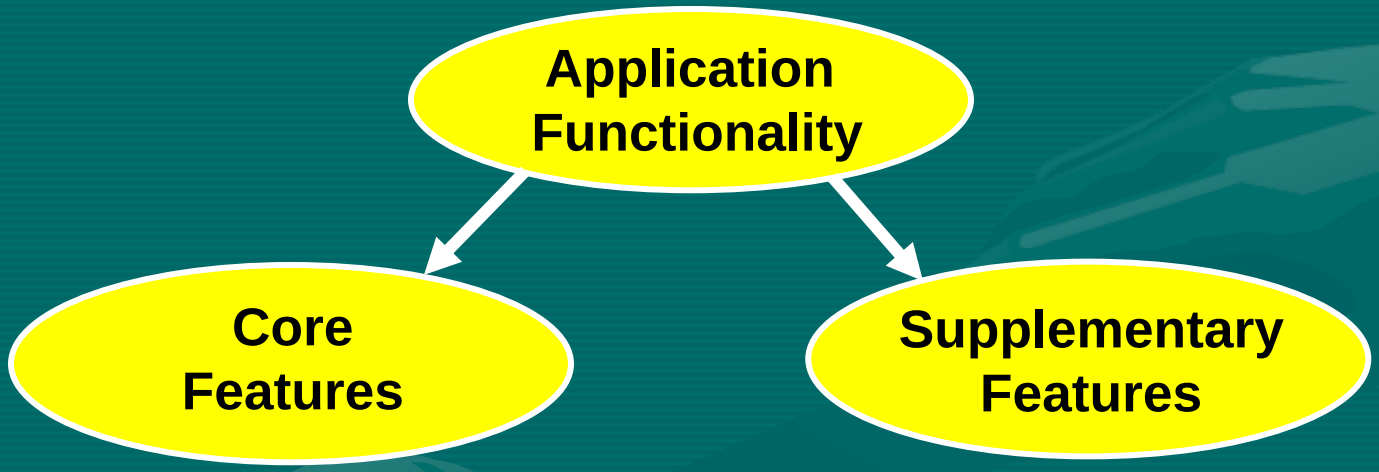
- It is a well-known fact that good software requirements are critical for the success of software projects.
- All emerging requirements methodologies had a goal to produce good software requirements.
- The **IEEE Std. 830-1998** "Recommended Practice for Software Requirements" defines characteristics of good requirements as follows:
  - Correct
  - Unambiguous
  - **Complete** (*means both a complete set and complete individual reqs*)
  - Consistent
  - Ranked for importance
  - Verifiable
  - **Modifiable** (*means maintainable requirements*)
  - Traceable



The main objective of AORE is improving requirements completeness and maintainability.

Two Categories of Requirements

- In the case of business applications we implement different functional features that can be of two categories:
  - ***core*** features, and
  - ***supplementary*** features



Many of the supplementary features can be scattered across the application and tangled with core features; in AORE they are called crosscutting concerns.

Example: Impact of Supplementary Features

Use Case: Check-in Guest

- **Preconditions**:
  - User has a privilege to check-in guest
  - Reservation has "Reserved" status
- **Main Course**:
  - User selects a guest to check-in
  - System opens guest's reservation screen
  - User completes/updates guest's stay data and saves reservation
  - System changes the reservation status to "In-House" and sends the reservation data to other systems

Validate user privilege

Validate reservation status

Validate front-end connectivity

Validate data entry

Validate concurrency

Validate front-end connectivity

Add system interface details

Issue with Requirements Completeness

- The meaning of the term “requirements completeness” may vary depending on whether a requirement is qualified as a core feature or a crosscutting concern:
  - a given core feature specification is not complete without analyzing and capturing details about how other scattered features are invoked and are affecting the core feature context.
  - a crosscutting concern specification is not complete without analyzing and specifying where it can be invoked and how it can affect related core features.

Up to 75% of software defects can be allocated to crosscutting concerns whose impact has not been analyzed and documented.

Issue with Requirements Maintainability

- Requirements maintenance means that requirements artifacts initially produced in previous releases are kept updated to be used in future releases.
- There are two main reasons to maintain requirements over time:
  - **Effective impact analysis of change requests (**most of change requests, on average 65%-85%, relate to the existing functionality);
  - **Lower cost of requirements development (**it is much cheaper to maintain the existing documents, as opposed to creating new ones as in practice most of changes relate to the existing features).

In practice, requirements from previous releases are commonly not maintained.

Symptoms & Root-Causes of Non-Maintainable Requirements

- The symptoms of non-maintainable requirements:
  - impact analysis of change requests is not performed, and
  - existing requirements are not kept updated and re-used in future releases. Instead, a project team keeps developing new requirements for each release.
- Root-causes of non-maintainable requirements:
  - poor structure and incompleteness of requirements models.

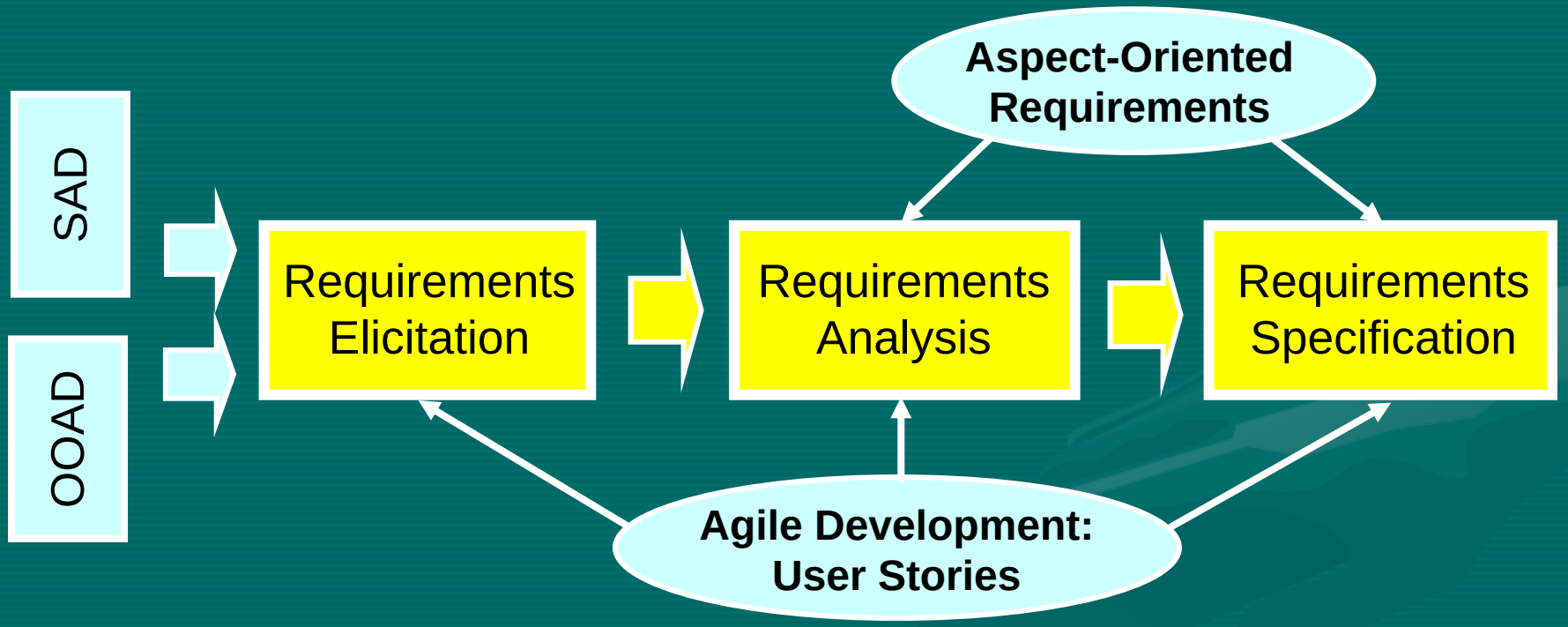
AORE specifically addresses the issue with requirements scattering and tangling and provides techniques to improve requirements completeness and maintainability.

AORE: Separation of Concerns

- AORE is a part of the general discipline known as Aspect-Oriented Software Development (www.aosd.net) that is based on the old software principle *separation of concerns*.
- The term “separation of concerns” (SoC) was coined in 1974 by Edsger Dijkstra in his article “On the role of scientific thought”. SoC means breaking a problem domain into specific aspects and then studying each aspect “in isolation for the sake of its own consistency”.

In AORE we apply the SoC principle to improve requirements structure and analysis.

AORE vs. Other Methodologies



- AORE is not a replacement for any of the existing methodologies.
- AORE offers requirements analysis and specification techniques that can be applied within any existing methodology to help us further improve requirements completeness and maintainability.

Part 1

AORE
Analysis Techniques

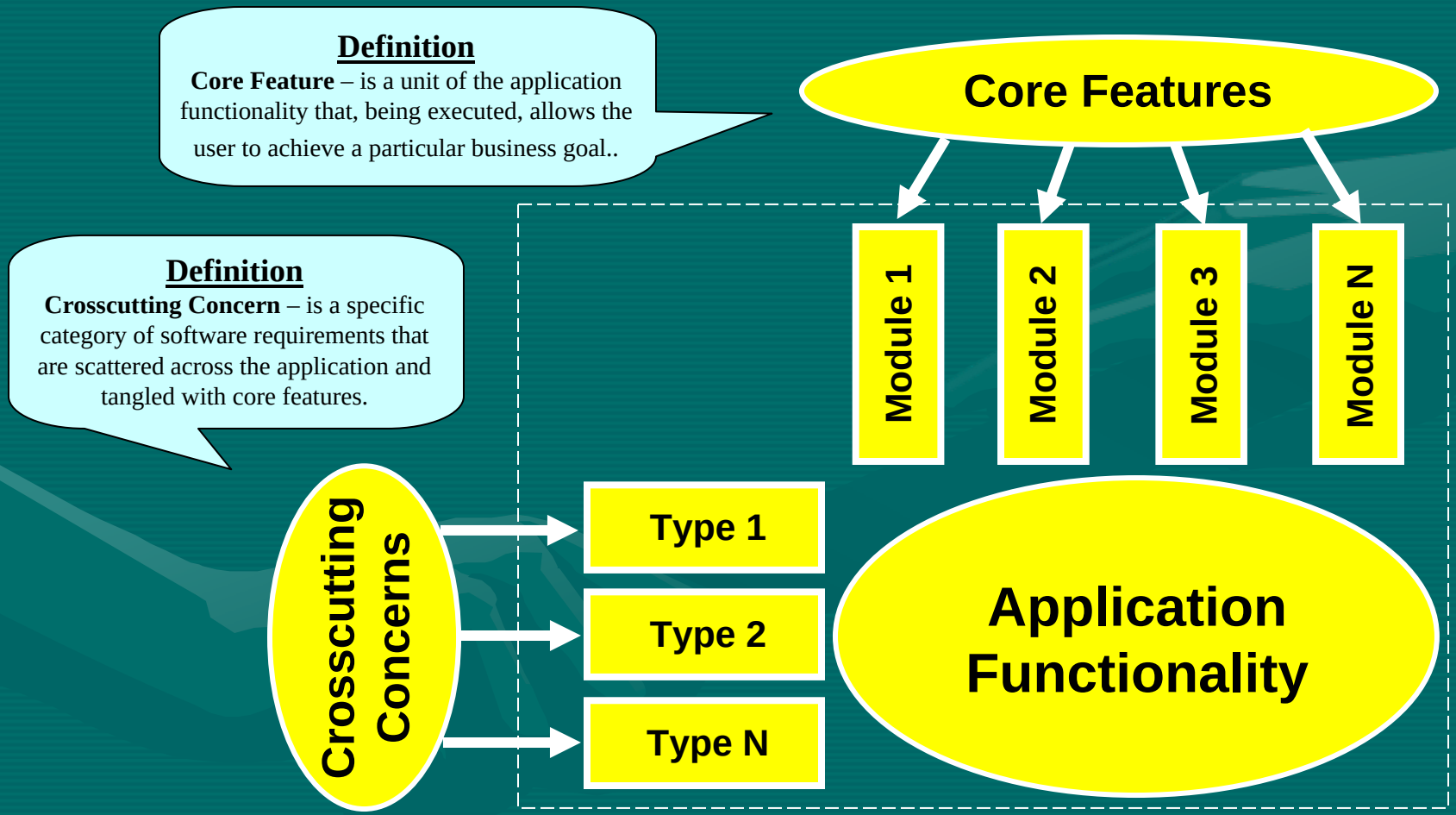
Analysis Phase: Traditional Application Decomposition

- Regardless of which methodology we follow, in the analysis phase we define modularization and structure of a future requirements model by:
  - decomposing the application into functional modules (a.k.a. functional areas), and
  - identifying the inventory of features (a.k.a. concerns);
- In SAD we perform top-down functional decomposition and identify a list of core features for each functional area.
- In OOAD we decompose functionality into use case packages and identify a list of use cases.

In SAD or OOAD the application functionality is partitioned along just one dimension, i.e., by core features.

AORE: Two-dimensional Decomposition

In AORE we follow the SoC principle and perform two-dimensional decomposition by **core features** and **crosscutting concerns**.



A complete inventory of concerns

Analysis Phase: Characteristics of Crosscutting Concerns

- Requirements that we can model as crosscutting concerns should have the following characteristics:
  - They cannot be invoked directly by end-users (i.e., they need a context of core features);
  - They impact the context of core features (i.e., they can constraint, interrupt, or enhance core features);
  - They are sufficiently scattered, i.e., they affect at least 2-3 core features;
- There some formal techniques to “mine” crosscutting concerns in the existing requirements documentation.
- In practice, we identify crosscutting concerns based on our prior experience with similar applications and our common sense.

Scattering and **tangling** are the main characteristics of crosscutting concerns.

Examples of Crosscutting Concerns

- The list of crosscutting concerns common to financial (e.g. trading) applications includes:

ET – Entitlements	DDV – Data Dependency Validation
AS – Account Setup	DDD – Data-Driven Defaults
RGN – Region	CN – Connectivity
PT – Product Type	CC – Concurrency
OT – Order Type (or Deal Type)	SI – System Interface
OS – Order Status (or Deal Status)	MB – Message Broadcasting
FV – Field Validation	CL – Calculations

See the meaning and descriptions of these crosscutting concerns in the Appendix section.

Analysis Phase: Analyzing Requirements Impact

- In AORE we analyze the mutual impact of requirements, in particular, how crosscutting concerns affect core features.
- The result of such analysis can be presented in the form of a Requirements Composition Table (RCT) that presents a holistic view of the application functionality.

RCT is an important deliverable of the Analysis phase when we follow the Aspect-Oriented Methodology.

Analysis Phase: Composing a Structure of Requirements

- We compose a structure of requirements by analyzing each core feature and making a decision about which of the crosscutting concerns affect the feature context, for example:

	Core Feature 1	Core Feature 2	.....	Core Feature N
Crosscutting Concern 1	1	1	1	1
Crosscutting Concern 2	1	0	1	1
.....	0	1	1	0
Crosscutting Concern N	1	1	0	1

1 – means applicable concern
0 – means not applicable concern

RCT Example (fragment): Hotel Management Application

"01. Front Desk" Module												
List of Concerns	**UC.01.01. Create Guest Reservation**	**UC.01.02. Update Guest Reservation**	**UC.01.03. Cancel Guest Reservation**	**UC.01.04. Check-In Guest**	**UC.01.05. Check-Out Guest**	**UC.01.06. Post Charges to Guest's Folio**	**UC.01.07. View, Update Folio Charges**	**UC.01.08. Create Message for Guest**	**UC.01.09. View, Cancel Message**	**UC.01.10. Add Travel Agency Commissions**	**UC.01.11. View, Update Travel Agency Commissions**	**UC.01.12. Manage Rooming List**
Core Functionality	1	1	1	1	1	1	1	1	1	1	1	1
GUI Features	1	1	1	1	1	1	1	1	1	1	1	1
Crosscutting Concerns												
ET - Entitlements	1	1	1	1	1	1	1	1	1	1	1	1
ST - Status	0	1	1	1	1	1	1	1	1	1	1	1
FV - Field Validation	1	1	0	1	0	1	1	0	0	1	1	1
DD - Data Dependency	1	1	0	1	0	1	1	0	0	0	0	0
CC - Concurrency	1	1	0	1	0	0	0	0	0	0	0	0
CN - Connectivity	1	1	1	1	1	0	0	0	0	0	0	0
SI - System Interface	1	1	1	1	1	0	0	0	0	0	0	0

1 – means applicable concern
0 – means not applicable concern

RCT Example (fragment): Equity Trading Application

List of Concerns	Modules																				
	01. Administration				02. Blotter Management				03. Alerts		04. Internal Order Management							05. External Order Management			
	01.01 Manage Expressions	01.02 Manage Scripts	01.03 Manage Trading Limits	01.04 Manage Rules	02.01 Manage Blotter Grid	02.02 Manage Groups	02.03 Apply Auto Filter	02.04 Export Data	03.01 Design Alerts	03.02 Evaluate Alerts	04.01 Route to Trading System	04.02 Enter Single Order	04.03 Enter Portfolio Order	04.04 Slice Order	04.05 Manage Client Coverage	04.06 Manage Portfolio (Merge, Delet	04.07 Roll Block Order to Next Day	05.01 Manage ATOMS Jar	05.02 Route Order for Execution	05.03 View, Handle Order Executions	05.04 Cross Oders
Core Functionality	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
GUI Features	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
Crosscuting Concerns																					
ET - Entitlements	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1
AS - Account Setup	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PT - Product Type	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1
OS - Order Status	1	1	1	1	0	0	0	0	1	1	1	0	0	1	0	0	1	0	1	1	1
FV - Field Validation	0	0	1	1	0	1	0	0	1	0	1	1	1	1	0	0	0	1	1	1	1
DD - Data Dependency	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	1
CN - Connectivity	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
CC - Concurrency	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	1	0	0
SI-Out - System Interface (to external trading systems)	0	1	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1	1	1	1
SI-In - System Interface (to external Upstream systems)	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0
MB - Message Broadcasting	1	1	1	1	0	1	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1
CL - Calculations	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	1

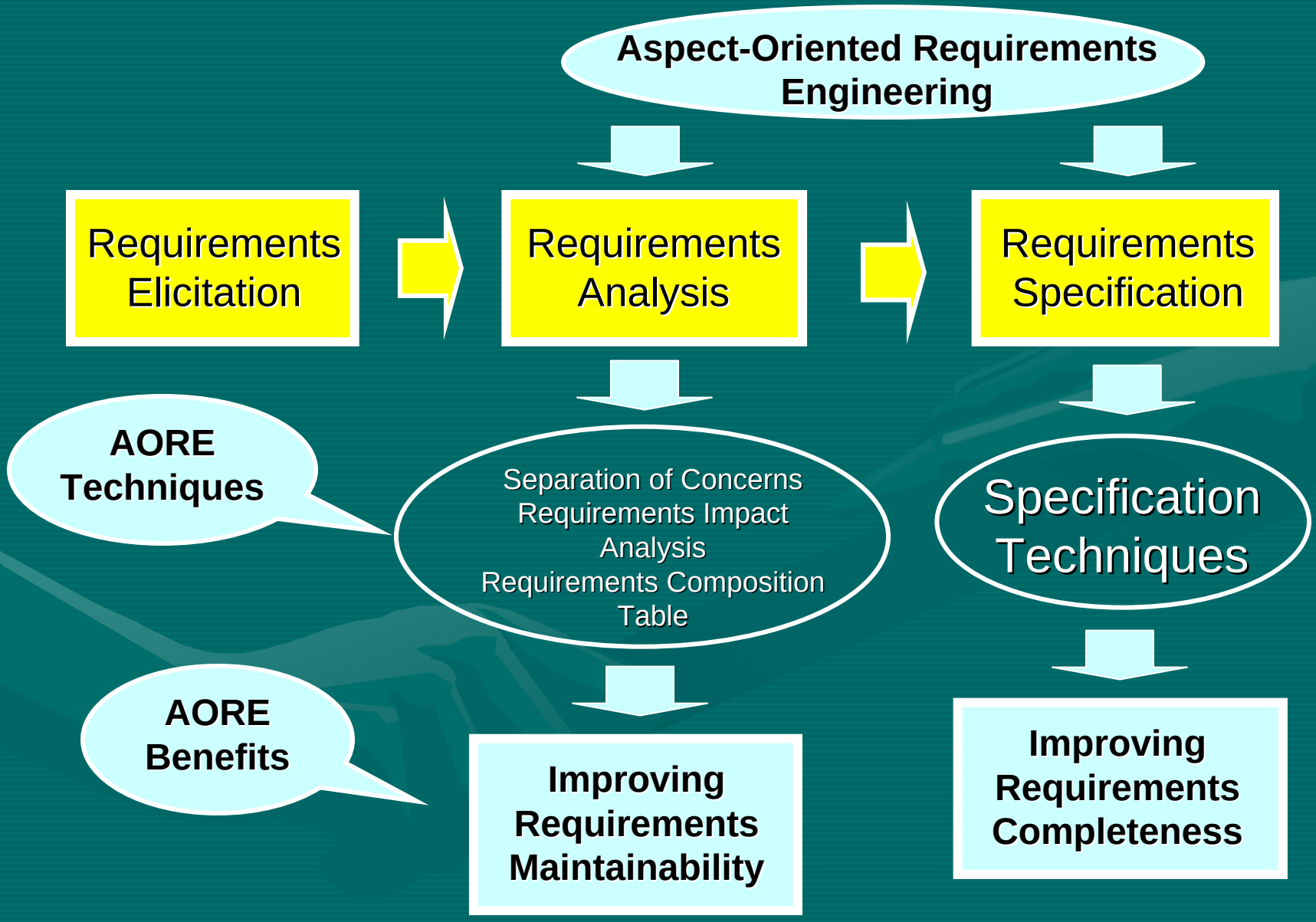
Requirements Composition Table vs. Requirements Traceability Matrix

- A Requirements Composition Table (RCT) should not be confused with the Requirements Traceability Matrix (RTM).
- An RCT is used to capture modularization and composition of requirements in order to provide a holistic view of the application functionality.
- An RTM's purpose is twofold:
  - To identify and maintain the relationship between software requirements and their sources, which is known as *backward traceability*;
  - to identify and maintain the relationship between requirements and other project artifacts developed based on requirements; this is known as *forward traceability*.

Summary of the RCT Benefits

- RCT provides a structured and holistic view of the application functionality and it can help us improve requirements analysis and maintainability.
- RCT is an important artifact of the Requirements Analysis phase that can effectively support various project tasks:
  - **Planning Iterative and Incremental Development**
  - **Supporting Agile Test-Driven Development**
  - **Requirements Reverse Engineering***
  - **Software Change Impact Analysis***
  - **Functional Test Planning**
  - **Test Coverage Assessment***
  - **Effective Exploratory Testing**
  - **Knowledge Transfer***

Summary of the AORE Techniques



Part 2

AORE Specification Techniques

Specification Phase: Defining Composition Rules

- The impact of crosscutting concerns can be classified by the following three cases:
  - Impose a constraint on a core feature
  - Interrupt a core feature flow and change its outcome
  - Add details to the affected core feature
- Based on this classification, AORE defines composition rules that can be used in functional specifications:
  - ***Wrap*** *(means imposing a constraint)*,
  - ***Override*** *(means interrupting a flow)*,
  - ***Overlap*** *(means adding details)*.
- In the **Analysis phase** we analyze and document the mutual impact of requirements in RCT.
- In the **Specification phase** we use the defined composition rules to specify that impact in requirements specifications.

Example: Agile User Story

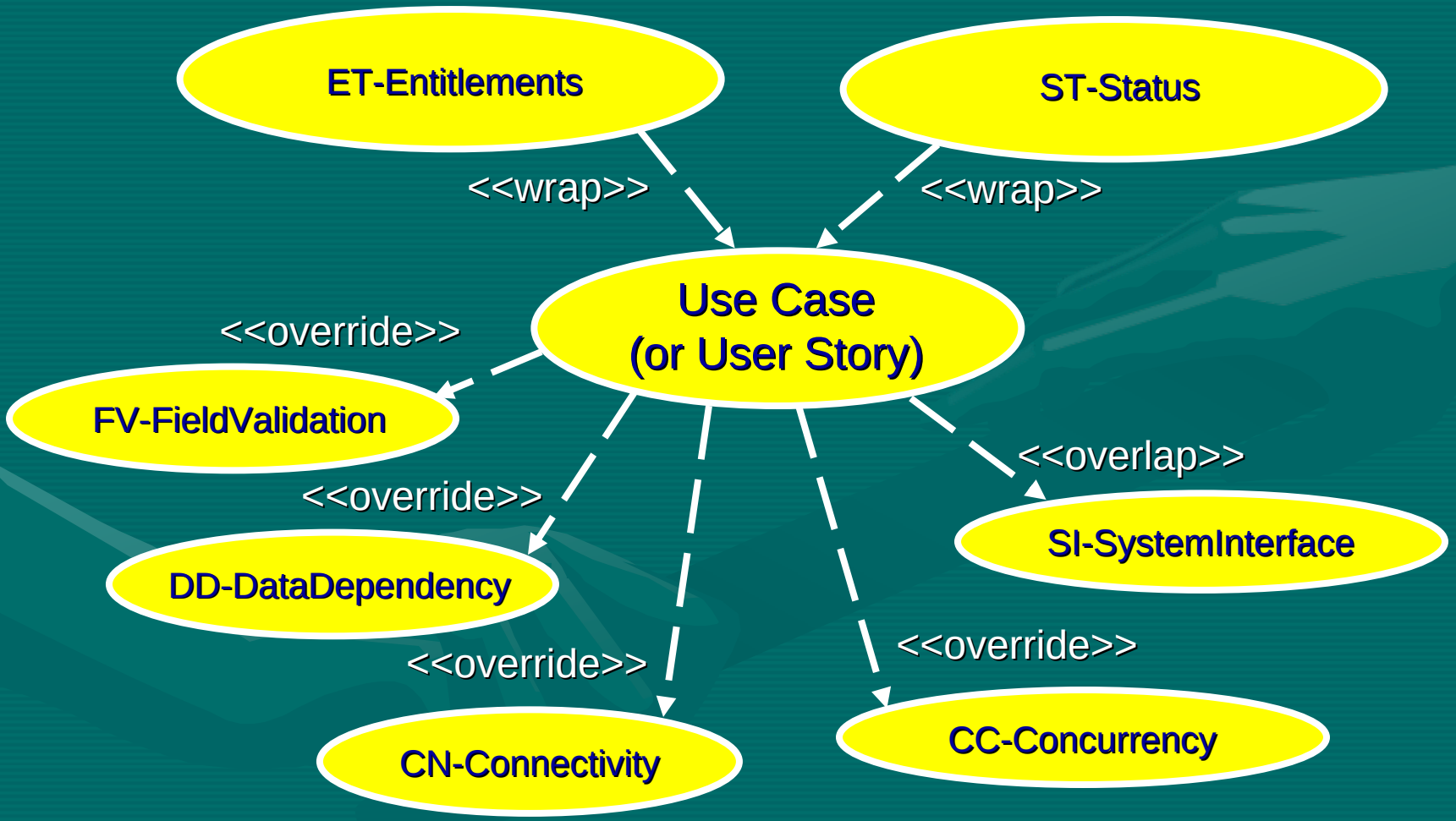
We will use an imaginary **Hotel Management System** to illustrate our discussion about composition rules.

User Story

“A user can check-in a guest who has a valid reservation.”

- From the Requirements Composition Table we already know which crosscutting concerns affect this story:
  - **ET-Entitlements** - Y
  - **ST-Status** - Y
  - **FV-Field Validation** - Y
  - **DD-Data Dependency** - Y
  - **CC-Concurrency** - Y
  - **CN-Connectivity** - Y
  - **SI-System Interface** - Y
- We then develop ideas about how these concerns affect the story context and select related composition rules (see next Slide).

Specification Phase: Composition Modeling with UML



Example: Documenting Realizations

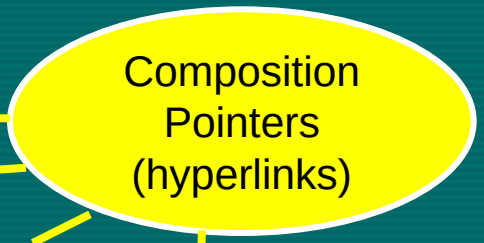
To see the **whole story**, we document realizations of the applicable crosscutting concerns:

- **ET-wrap** – to begin this story, the system validates user entitlements;
- **ST-wrap** – to begin this story, the system validates the reservation status that should be "Reserved";
- **FV-override** – to complete this story, the system checks whether the reservation's data-entry fields have valid values;
- **DD-override** – to complete this story, the system checks whether some fields have valid combinations (e.g. Arrival Date < Departure Date, Departure Date < Credit Card Exp. Date);
- **CN-override** – to complete this story, the system checks whether the front-end stays connected;
- **CC-override** – to complete this story, the system checks whether the selected room is not already taken by another front-desk clerk;
- **SI-overlap** – to complete this story, the system sends the reservation data to the Central Reservation System and Guest Rewards System.

By specifying **realizations** of crosscutting concerns we add necessary details to the story context.

Example: Use Case Scenario

UC.01.04 Check-in Guest
Pre-conditions:
User has a privilege to check-in guest (**ET-wrap1**).
Guest's reservation has a status "Reserved" (**ST-wrap2**)



Normal Course of Events:

1. This use case starts when User selects Guest' name from the "Today's Arrival" list to access her reservation.
2. System displays Guest's reservation data and changes the reservation status from "Reserved" to "In-House" (**CN-override1**).
3. User verifies/updates Guest's reservation data.
4. User selects a room for the guest.
5. User verifies/updates the guest's payment information.
6. User completes the check-in process and submits the reservation.
7. System validates the check-in information, sends the reservation data to other systems, and displays a check-in confirmation message (**FV-override2**, **DDV-override3**, **CC-override4**, **CN-override5**, **SI-overlap1**)
8. User acknowledges the check-in confirmation.
9. System brings User back to the main screen, removes Guest's name from "Today's Arrivals" list, and this use case ends.

Join Points and Composition Pointers

Join Point Definition

- A join point indicates where the impact of crosscutting concerns is inserted.
- A join point could be either a use case precondition or use case step that is impacted by one or more crosscutting concerns.

Composition Pointer Definition

- A composition pointer indicates what and how affects a give step in the use case scenario and is included in the affected join point.
- Each composition pointer, for example (ET-wrap1) or (FV-override2), is composed of the crosscutting concern type and its composition rule and it serves a dual purpose:
  - First, it indicates what and how impacts a given join point, so we could clearly see what else we need to consider at this point in the use case scenario.
  - Second, it is used as a hyperlink to quickly navigate to details of a given crosscutting concern realization documented and bookmarked in a separate section “Appendix” of a use case specification.

Example: Documenting Realizations

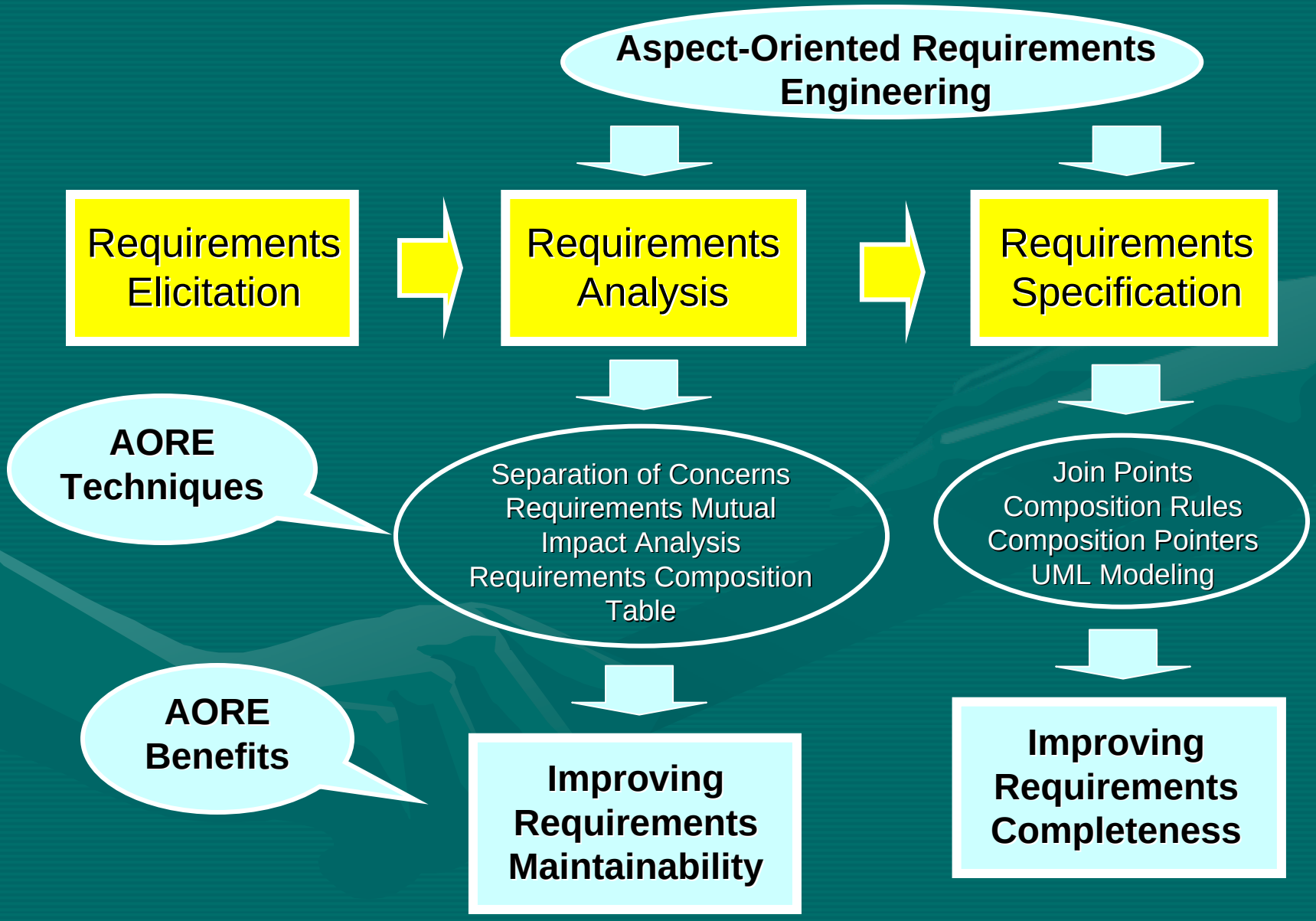
Use Case Appendix: Realizations of Crosscutting Concerns

Composition Pointers	Realizations of Crosscutting Concerns	References to Supplementary Requirements
ET-wrap1	This use case can be executed only by the following user roles: ▪General Manager ▪Front Desk Manager ▪Front Desk User For other roles this feature is not available.	SR_ET: 01.05
ST-wrap2	This use case can be executed only when a reservation status is "Reserved". For other reservation statuses this feature is not available.	SR_ST: 01.01
CN-override1	System checks for the front-end connectivity before opening the guest's reservation.	SR_CN: 01.01
FV-override2	System validates individual fields: Guest Name Guest Address Number of Nights, etc.	SR_FV: 01.03 01.09 01.12
DDV-override3	System validates a combination of fields: Check-in Date < Check-out Date; Check-out Date < Credit Card Expiration Date;	SR_DDV: 01.04 01.05
CC-override4	System validates concurrent selection of the same room for different guests.	SR_CC: 01.01
CN-override5	System checks for the front-end connectivity before saving a reservation.	SR_CN: 01.02
SI-overlap1	System sends the check-in information to the Central Reservation System and Guest Rewards System.	SR_SI: 01.02 01.08

Composition pointers (bookmarks)

References to realization details in supplementary requirements.

Summary of the AORE Techniques



Benefits of the AORE Specification Techniques

- AORE can help us improve requirements completeness regardless of what specification style we use – traditional style, use cases, or user stories.
- AORE defines join points and composition rules to document the impact of crosscutting concerns in the context of core features, thus improving requirements completeness.
- AORE provides guidelines to model the composition of concerns with UML.

Part 3

Using RCT for Change Impact Analysis

Change Impact Analysis – Why?

Common Context

- Most of requested changes (on average 65% - 85%) overlap with the existing functionality.
- Commonly, there is a many-to-many relationship between change requests and impacted existing features.
- Complete understanding of the change impact is necessary for accurate effort estimation and producing a quality software.

Common Issues

- There is no complete requirements model, at best requirements are managed only by release-specific slices of functionality.
- When a complete requirements model is not available, performing impact analysis is very difficult.
- Lack of understanding the impact of changes is the main reason for poor quality of software products and schedule overruns.

Change Impact Analysis – How?

Use of RCT

- A Requirements Composition Table can be effectively used as a frame of reference to analyze and document the impact of changes.
- When there is no complete requirements model, reverse-engineer the existing functionality and produce a Requirements Composition Table.

Requirements Reverse-Engineering Steps

1. Conduct a kickoff meeting with a development manager and BA to agree on functional decomposition and list of crosscutting concerns.
2. Conduct interview sessions with project team members to identify features for each of the functional areas and produce an RCT.
3. Refine the list of crosscutting concerns, update the RCT.

To complete reverse-engineering of a mid-size application and produce an RCT takes on average 7-10 hours.

Approach to Performing Change Impact Analysis

Change Impact Analysis Guidelines

- Understand the type of change:
  - Behavior-related, analyze what kind of behavior:
    - Core feature, or
    - Crosscutting concern
  - Data-related, analyze:
    - Where the data change should be implemented
    - Where the changed data will be used across the application (e.g. core features, reports, etc)
- Capture the impact analysis results in RCT
- Analyze changes applied to the same existing core feature to see whether they can conflict with each other

Example: Documenting the Results of Change Impact Analysis

Front Office Module																						
Concern Types	UC.1.01 Create New SH Guest Reservation	UC.1.02 Update SH Guest Reservation	Reservation	any Account	Account	ccount	UC.1.0	UC.1.09 Check-Out Guest	UC.1.10 Post New Charges to Folio	UC.1.11 View, Update Folio Charges	UC.1.12 Quick Posting of Folio Charges	UC.1.13 Create New Message	UC.1.14 View, Cancel Message	UC.1.15 Create New Group	UC.1.16 View, Update Group	UC.1.17 Cancel, Close Group	UC.1.18 Create New CRS Guest Reservation	UC.1.19 Update CRS Guest Reservation	UC.1.20 Cancel CRS Guest Reservation	UC.1.21 Create, Update Company Statement	UC.1.22 Add, View Travel Agency Commissions	UC.1.23 Block Room for Company	UC.1.24 Manage Rooming List
Core Functionality	BRD.4.1.24	BRD.4.1.6 BRD.4.1.24	1	1	1	1	1	BRD.4.4.5 BRD.4.1.24	BRD.4.1.17 BRD.4.4.1	BRD.4.4.1	BRD.4.1.17 BRD.4.4.1	1	1	1	1	1	1	BRD.4.1.6	1	1	1	1	1
GUI - User Interface	BRD.4.1.10	BRD.4.1.6 BRD.4.1.10	BRD.4.1.10	1	0	0	0	0	BRD.4.1.4 BRD.4.1.26 BRD.4.4.3	BRD.4.4.3	BRD.4.1.4 BRD.4.1.26	1	0	1	0	0	0	BRD.4.1.6	0	1	1	1	1
Crosscutting Concerns																							
FV - Field Validation	BRD.4.1.26	BRD.4.1.9 BRD.4.1.26	0	1	1	0	1	0	BRD.4.4.3	0	1	1	0	1	1	0	1	BRD.4.1.9	1	1	1	1	1
DD - Data Dependency	1	1	0	0	0	0	1	0	1	0	1	1	1	0	1	1	1	1	0	1	0	1	0
CC - Concurrency	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
SI - System Interface	1	1	1	1	1	1	1	1	BRD.4.4.3	BRD.4.4.3	0	0	0	1	1	1	1	1	1	0	0	0	0
CN - Connectivity	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
RP - Reports	1	1	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0
ST - Status	0	1	1	0	1	1	1	1	1	BRD.4.1.21	0	1	1	1	1	1	0	0	0	1	1	0	1
DF - Data Flow	BRD.4.4.7	BRD.4.4.7					BRD.4.4.7																

Impact of Changes

When multiple change requests affect the same existing feature they can conflict with each other.

Benefits of Change Impact Analysis

- A Development Manager can better see the effort to implement changes and better estimate the schedule.
- A Development Team can produce code of better quality.
- A QA Team can better understand quality risks, better plan functional testing, and design more complete tests.

Part 4

Using RCT for Test Coverage Assessment

Reference:
Y. Chernak "Mind the Gap: Using a Requirements Composition Table to Assess Test Coverage"
Better Software, March 2008

Common Test Process Issue

- On critical projects testers commonly maintain and execute a regression test suite. In this case we are concerned with the test suite completeness, as our final application certification is only as good as the regression suite coverage.
- A conventional technique to measure test coverage requires complete software requirements to be used as a frame of reference.
- Common Issue:
  - Testers in the field do not have complete requirements, as a result, they do not have visibility into regression suite completeness and test coverage gaps.
  - Statistics show that on such projects test suites are commonly incomplete with the average coverage between 15 % to 35 %.

Purposes of the RCT Technique

- In test coverage assessment the RCT technique is primarily intended to support our analysis from three perspectives:
  1. identifying test coverage gaps, i.e., requirements not covered by tests;
  2. providing testers with visibility into which requirements have more and which ones have less test coverage;
  3. helping testers understand which types of concerns they commonly miss in test designs;
- Note, assessment of test coverage is not the same as evaluation of test design completeness for individual requirements. For example, you can create tests for all application features and achieve complete test coverage; still some of the individual requirements can lack necessary test cases.

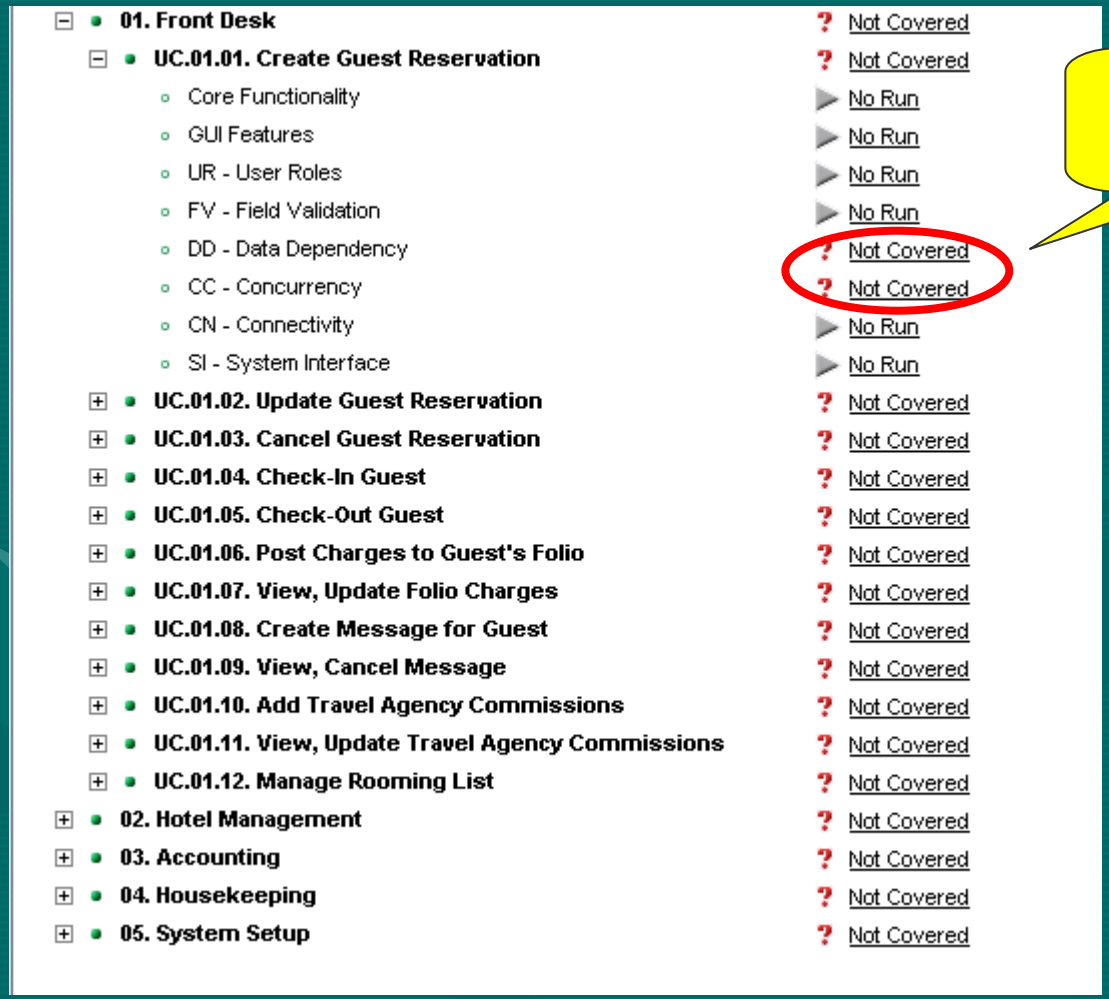
The Assessment Steps

- The RCT technique to assess test coverage is composed of the following three steps:
  1. Reverse-engineer requirements and present their structure in a requirements composition table (*we discussed in Part 1 and Part 3*)
  2. Establish traceability between requirements and regression tests
  3. Measure test coverage and identify gaps

Step 2: Establish Traceability

- Test management tools, like HP Quality Center (QC), can be very effective in establishing traceability between requirements and tests.
- First, create in QC a requirements repository following the same requirements structure as we created in the RCT.
- Second, use this inventory and structure of requirements as a frame of reference to establish traceability from the tests to related requirements.
- Depending on your test design, it is possible that a given test can be traced to more than one concern. Continue this activity until all regression tests are traced to their related requirements.

Example: Requirements Repository in HP Quality Center



Tip:
Create the test repository structure the same as requirements, then maintaining tests and managing traceability between requirements and tests will be more efficient.

Step 3: Measure Test Coverage and Identify Gaps

- To derive coverage measurements, mark with a different color the RCT cells that represent requirements not covered by tests.
- After that derive coverage measurements from two perspectives:
  - a) test coverage for each core feature context, and
  - b) test coverage for each concern type (across all core features).
- Measure test coverage for each core feature context (i.e. a column in the RCT) as follows:
  1. calculate the sum of all applicable concerns for a given core context. This number represents 100% of concerns to be covered by tests;
  2. subtract from the sum the number of colored cells, i.e., concerns not covered by tests;
  3. calculate the ratio of covered concerns to the total number of concerns and present it as a percentage of requirements covered by tests;
  4. Calculate coverage for each module and for the entire application.

Step 3 (cont'd): Presenting the Test Coverage Results

- When we execute regression testing we always have limited time. This means that a test team should decide which of the software requirements are most important to validate and agree on the regression test scope.
- Commonly, validating core functionality is a minimal requirement for regression test coverage. In addition, testers might decide that some of the identified crosscutting concerns should be included in the regression test scope.

The assessment results can be presented as a range between covering only the core functionality and covering a complete inventory of requirements, including all crosscutting concerns.

Example 1: Assessment Results in RCT

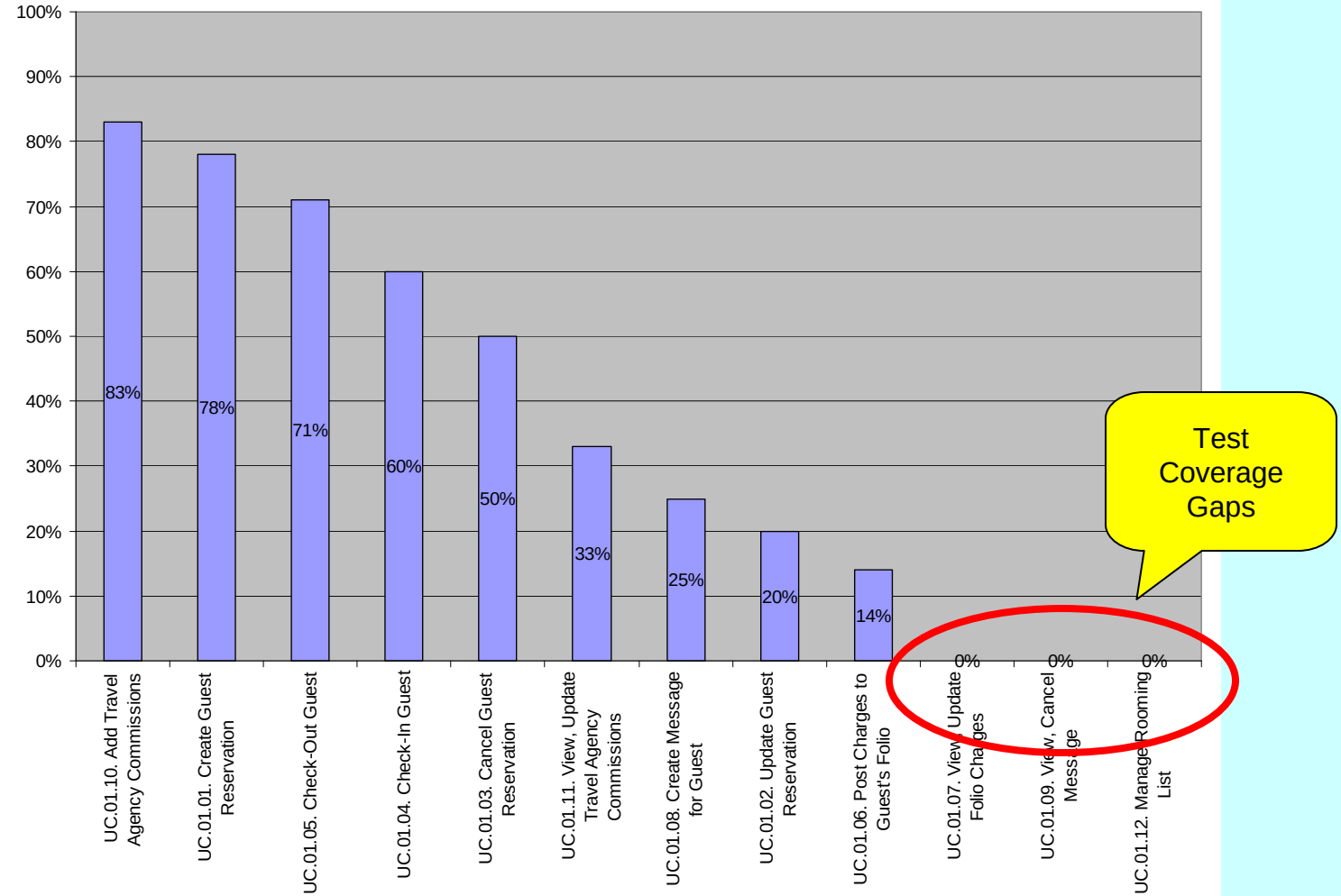
"01. Front Desk" Module - Test Coverage Analysis

Testing Concerns	UC.01.01. Create Guest Reservation	UC.01.02. Update Guest Reservation	UC.01.03. Cancel Guest Reservation	UC.01.04. Check-In Guest	UC.01.05. Check-Out Guest	UC.01.06. Post Charges to Guest's Folio	UC.01.07. View, Update Folio Charges	UC.01.08. Create Message for Guest	UC.01.09. View, Cancel Message	UC.01.10. Add Travel Agency Commissions	UC.01.11. View, Update Travel Agency Commissions	UC.01.12. Manage Rooming List	Coverage by Concern Types
Core Functionality	1	1	1	1	1	1	1	1	1	1	1	1	75%
GUI Features	1	1	1	1	1	1	1	1	1	1	1	1	42%
Crosscutting Concerns													
ET - Entitlements	1	1	1	1	1	1	1	1	1	1	1	1	25%
ST - Status	0	1	1	1	1	1	1	1	1	1	1	1	18%
FV - Field Validation	1	1	0	1	0	1	1	0	0	1	1	1	25%
DD - Data Dependency	1	1	0	1	0	1	1	0	0	0	0	0	0%
CC - Concurrency	1	1	0	1	0	0	0	0	0	0	0	0	0%
CN - Connectivity	1	1	1	1	1	0	0	0	0	0	0	0	60%
SI - System Interface	1	1	1	1	1	0	0	0	0	0	0	0	100%
Test Coverage by Use Cases:	75%	22%	50%	56%	83%	17%	0%	25%	0%	80%	40%	0%	37%

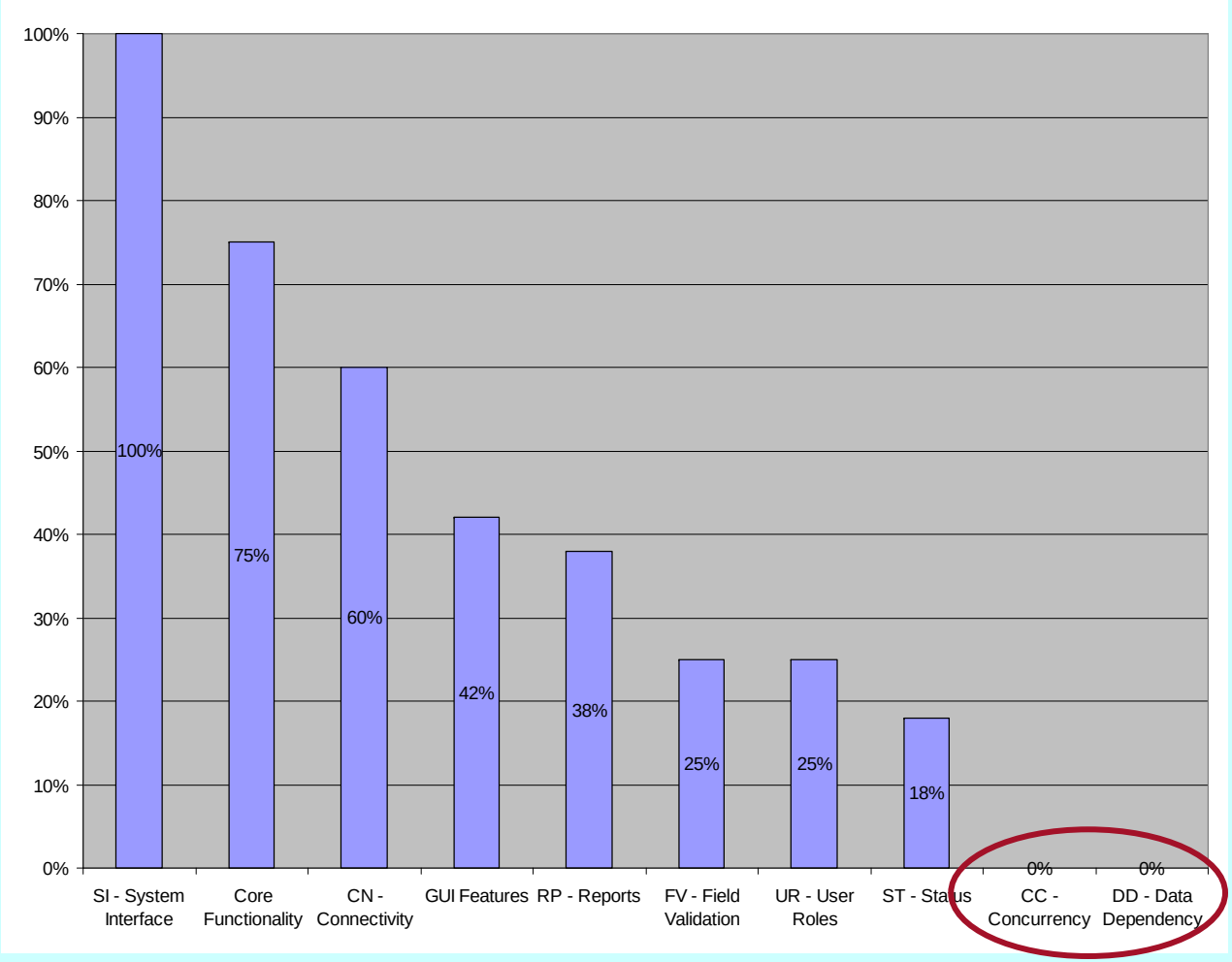
0 - means not applicable concern
1 - means applicable concern
Yellow-colored Cell - means missing tests (gap)

Test coverage Range: 37% - 75%

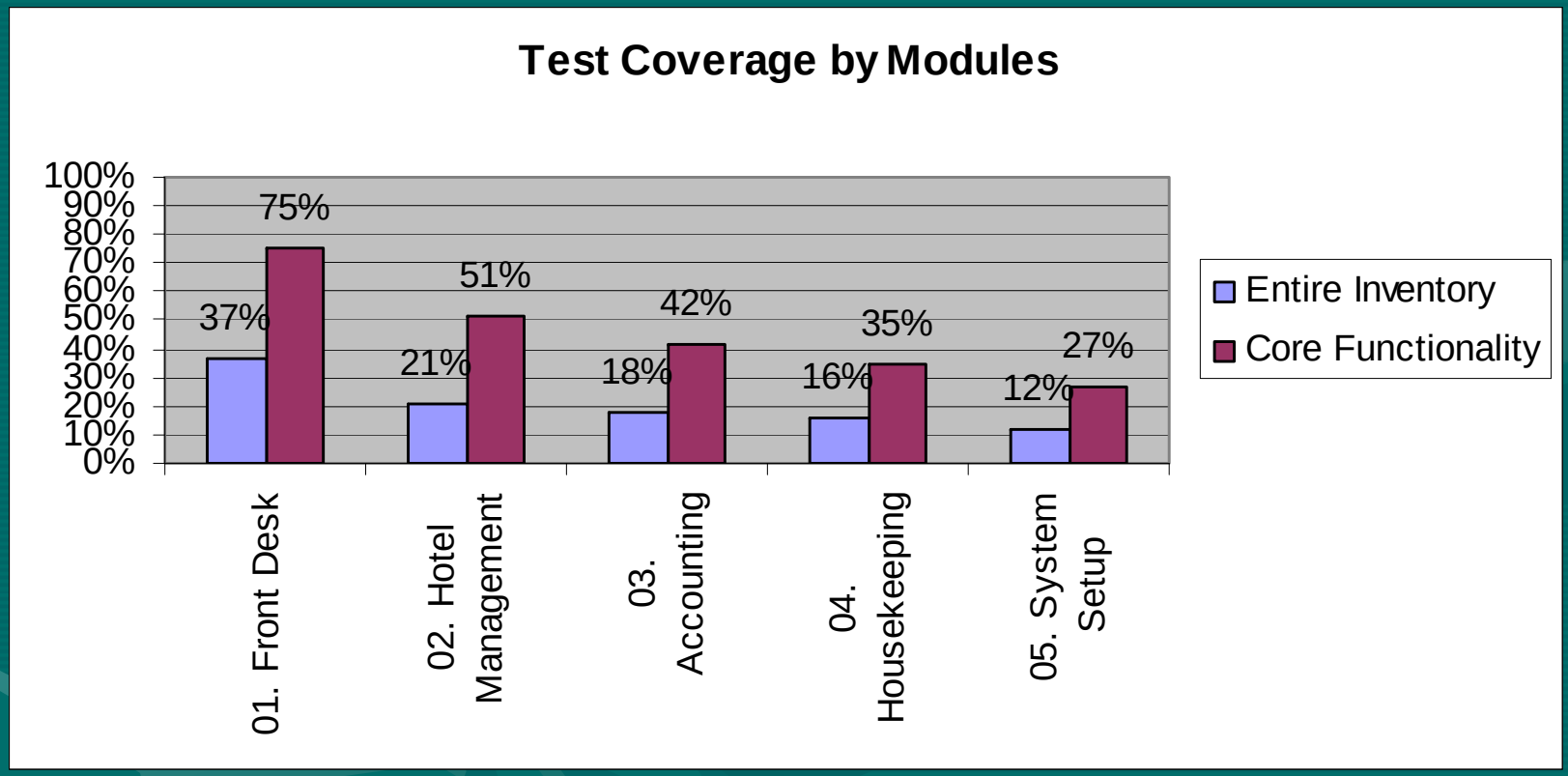
Example 2:
Assessment Results by Core Features



Example 3:
Assessment Results by Concern Types



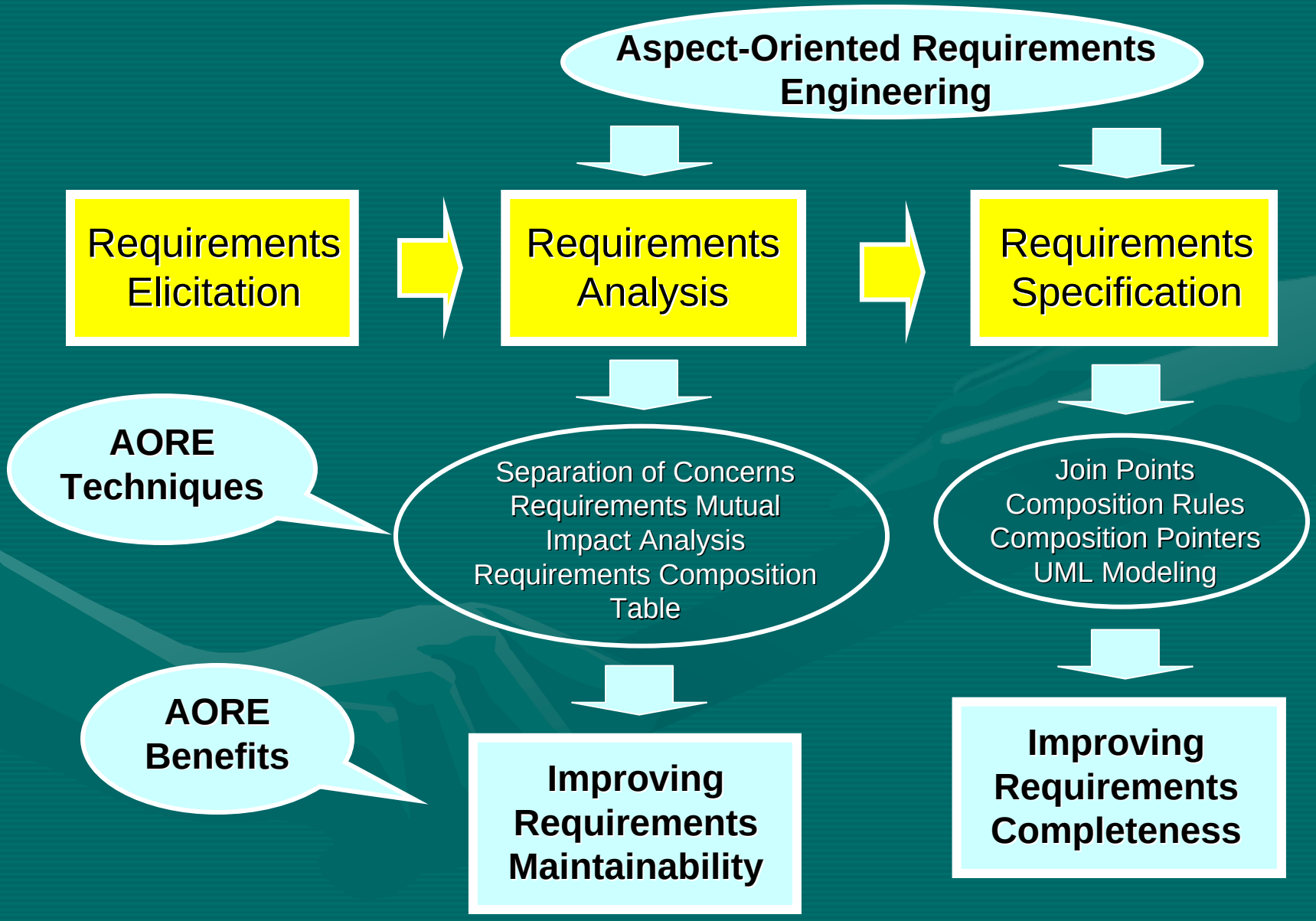
Example 4: Assessment Results by Modules



The assessment results for each module are presented as a range. A QA team should decide what test coverage is really necessary.

Presentation Summary

Summary of the AORE Techniques



Conclusion

- AORE does not replace, but rather complements any of the existing requirements methodologies.
- AORE provides techniques to develop better-structured requirements that can help us improve requirements completeness, maintainability, and cost of development.
- AORE is a young methodology that is still evolving, exploring its benefits, and making its way to practitioners.
- AORE current challenges:
  - consolidating various composition rules into a standard set of rules;
  - adopting standard UML techniques to support aspect-oriented requirements modeling;
  - reconciling the differences between the two schools, i.e., two-dimensional vs. multi-dimensional separation of concerns to form a consistent methodology.

Appendix A

Descriptions of
Crosscutting Concerns Common
to Financial Applications

Descriptions of Crosscutting Concerns

- Crosscutting concerns common to business applications:

Concern Type	Description
ET—Entitlements	This concern relates to defining various user entitlements and specifying which core features can be executed by a given entitlement.
AS—Account Setup	This concern relates to specifying attributes of client account setup and how they affect behavior of core features.
RGN—Region	This concern is common to global financial applications that are used in different regions around the globe. It relates to specifics of core feature behavior that depends on the different regions.
ST—Status	This concern specifies a lifecycle of a particular entity—for example, a reservation in the case of a hotel management application or a trade order in the case of trading applications. A lifecycle is commonly composed of various statuses that affect and constrain execution of core features. For example, if a reservation has a status “Canceled,” a user cannot check in or check out a guest. If a company status is “Inactive,” a user cannot add or update travel agency commissions for this company.
PT—Product Type	This concern is common to trading systems where a given application can be used for trading different financial products. For example, an equity-trading application can be used for trading stocks, options, and indices. Depending on a product type, some core features can behave differently.

Descriptions of Crosscutting Concerns (cont'd)

Concern Type	Description
FV—Field Validation	This concern is common to any business application and relates to validating individual data entry fields.
DDV—Data Dependency Validation	This concern is common to any business application and relates to validating a combination of related fields. For example, in the case of a hotel management system, a reservation's check-in date should be before a check-out date, the check-out date should be before a credit card expiration date, etc.
CN—Connectivity	This concern is common to any business application that is composed of different components communicating over a network. It relates to validating that the system's front end stays connected while the user completes a given transaction, and it defines the alternative behavior when the application goes into a disconnected state.
CC—Concurrency	This concern is common to any multi-user business application. It relates to handling concurrent manipulation of the same data by multiple users.
SI—System Interface	This concern is common to any business application that has interfaces to external systems. It relates to details of sending and receiving data to or from other systems (upstream or downstream). Such a concern can affect many core features that either use data from external systems, or produce and send data to external systems.