PACE UNIVERSITY
Seidenberg School of Computer Science and Information Systems

CS631G Software Verification

ASPECT-ORIENTED REQUIREMENTS ANALYSIS (W4)

Class instructor: Yuri Chernak, PhD

Outline

Part I. Introduction

- Challenges to produce complete and maintainable requirements
- The principle of "separation of concerns"
- Aspect-oriented classification of requirements
- History, objectives, and techniques of the AORE
- AORE vs. other requirements methodologies

Part II. AORE Analysis

- The concept, characteristics, and examples of crosscutting concerns
- The purpose and benefits of a Requirements Composition Table (RCT)
- Steps to produce an RCT's main view
- RCT ownership and maintenance

Part III. RCT-based Change Impact Analysis

- The importance of change impact analysis (CIA)
- CIA process phases and roles
- Release management and CIA

Key Points

Appendix A: Crosscut Analysis Examples

Appendix B: RCT's frequently asked questions

Appendix C: Examples of crosscutting concerns

Class Objectives

The objective of this class is to introduce a new requirements discipline - Aspect-Oriented Requirements Engineering (AORE), with a focus on requirements analysis techniques.

At the end of this class you will:

- understand the issues with requirements completeness and how the AORE methodology can help us develop better requirements;
- understand the concept of crosscutting concerns;
- learn the Requirements Composition Table (RCT) technique and steps to develop an RCT;
- learn an RCT-based approach to performing change impact analysis.

Part I.
Introduction to Aspect-Oriented Requirements Engineering (AORE)

Reference

Moreira, R. Chitchyan, J. Araújo, A. Rashid, *Aspect-Oriented Requirements Engineering*, Springer-Verlag, 2013

Characteristics of Good Requirements

The **IEEE Std. 830-1998** "Recommended Practice for Software Requirements" defines characteristics of good requirements:

- Correct
- Unambiguous
- Traceable
- Consistent
- Ranked for importance
- Verifiable
- **Modifiable** (i.e., maintainable)
- **Complete**

The most challenging characteristics to implement

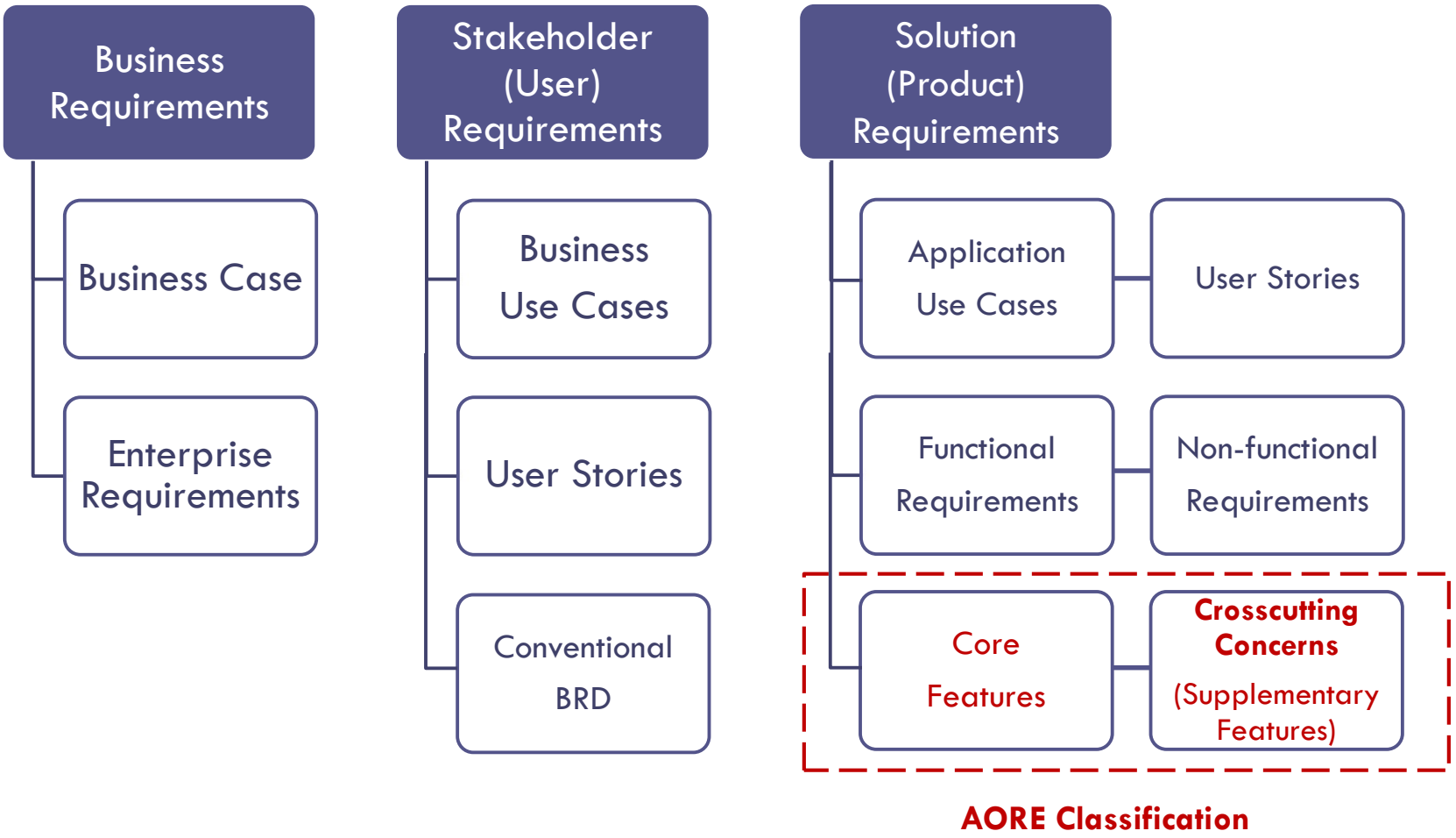
The Aspect-Oriented methodology focuses on improving requirements **completeness** and **maintainability.**

Software Principle – "Separation of Concerns"

- AORE is a part of the general discipline — Aspect-Oriented Software Development, which is based on the old software principle known as the Separation of Concerns (SoC).
- The term "separation of concerns" 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 analyze an application's functionality from two perspectives – *core features* and *crosscutting concerns*.

BABOK Types of Requirements



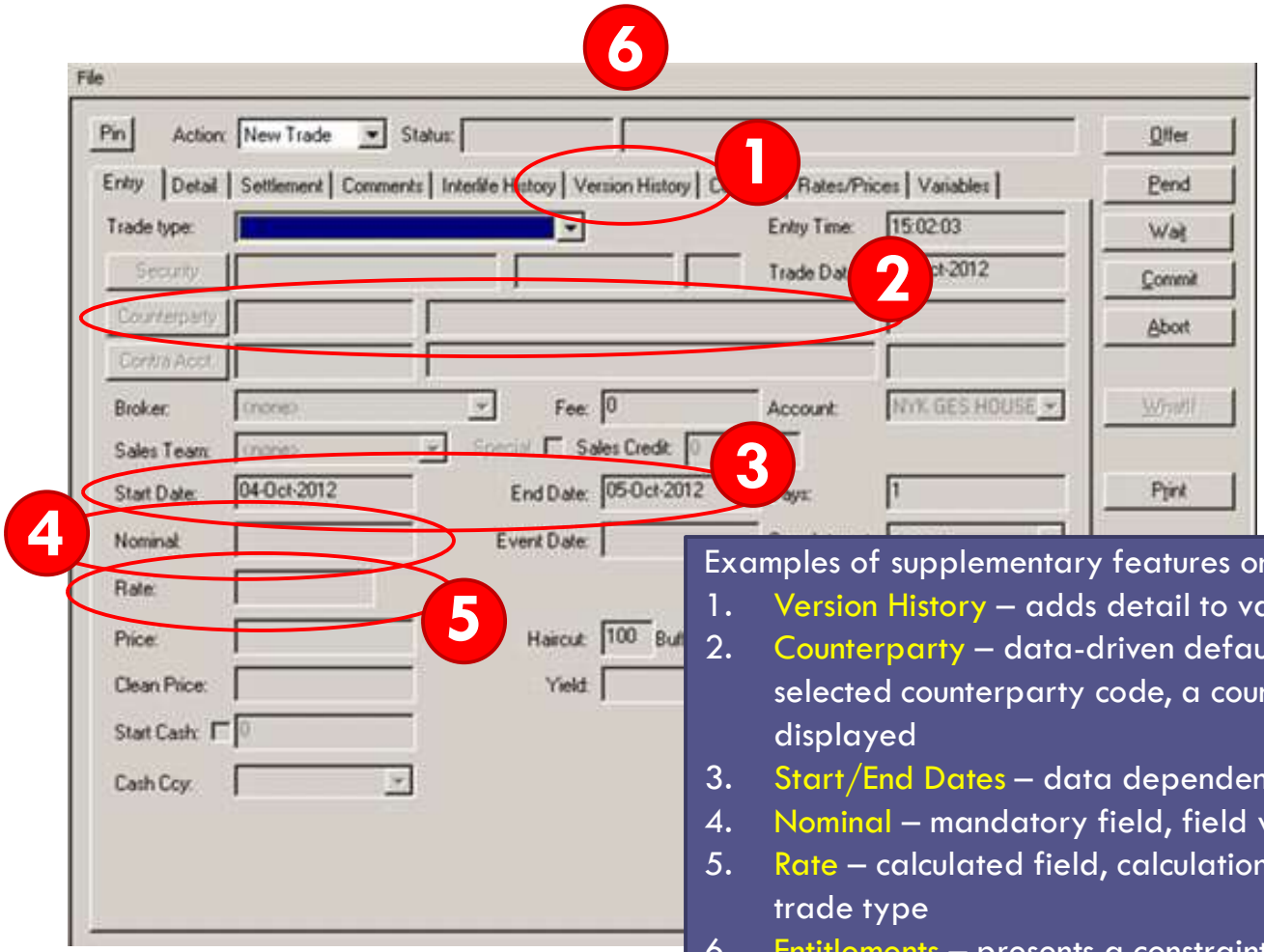
*) *BABOK – business analysis body of knowledge*

Core vs. Supplementary Features

Core Features	• features that, being executed by the end-user, produce a distinct and tangible business result.
Crosscuts (Supplementary Features)	• features that do not produce business results by themselves, but capture some necessary details that can be added to the context of core features.

- Commonly, supplementary features can be scattered across the application and tangled within the context of various core features.
- When this happens, such features are called in the AORE — Crosscutting Concerns (a.k.a. Early Aspects).

Example: Impact of Supplementary Features



Core features using this screen:

- New Trade
- View Trade
- Cancel Trade
- Correct Trade
- Close Trade

Examples of supplementary features on this screen:

1. Version History – adds detail to various core features
2. Counterparty – data-driven defaults, depending on the selected counterparty code, a counterparty description is displayed
3. Start/End Dates – data dependency validation
4. Nominal – mandatory field, field validation is applied
5. Rate – calculated field, calculation is applied depending on a trade type
6. Entitlements – presents a constraint, not every user can use this screen to create a new trade or cancel a trade

Example: Statistics from Trading Application

- On average, each core feature of the example trading application is impacted by and tangled with ten other supplementary features.
- Examples of the most-scattered crosscutting concerns:

Crosscut Category	Impacted Core Features %
Entitlements	77 %
Data Flow In	70%
Client Setup	65%
Data Flow Out	57%
Data Enrichment	53%
Field Validation	51%

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 **supplementary** feature:

- A given core feature specification is not complete without analyzing and capturing details about how scattered supplementary features, associated with it, are invoked and how they affect the core feature's context.
- A supplementary feature specification is not complete without analyzing and specifying where it can be invoked and how it can affect related core features.

Classification of product requirements by core features and crosscutting concerns can help us better analyze requirements and produce more complete specifications.

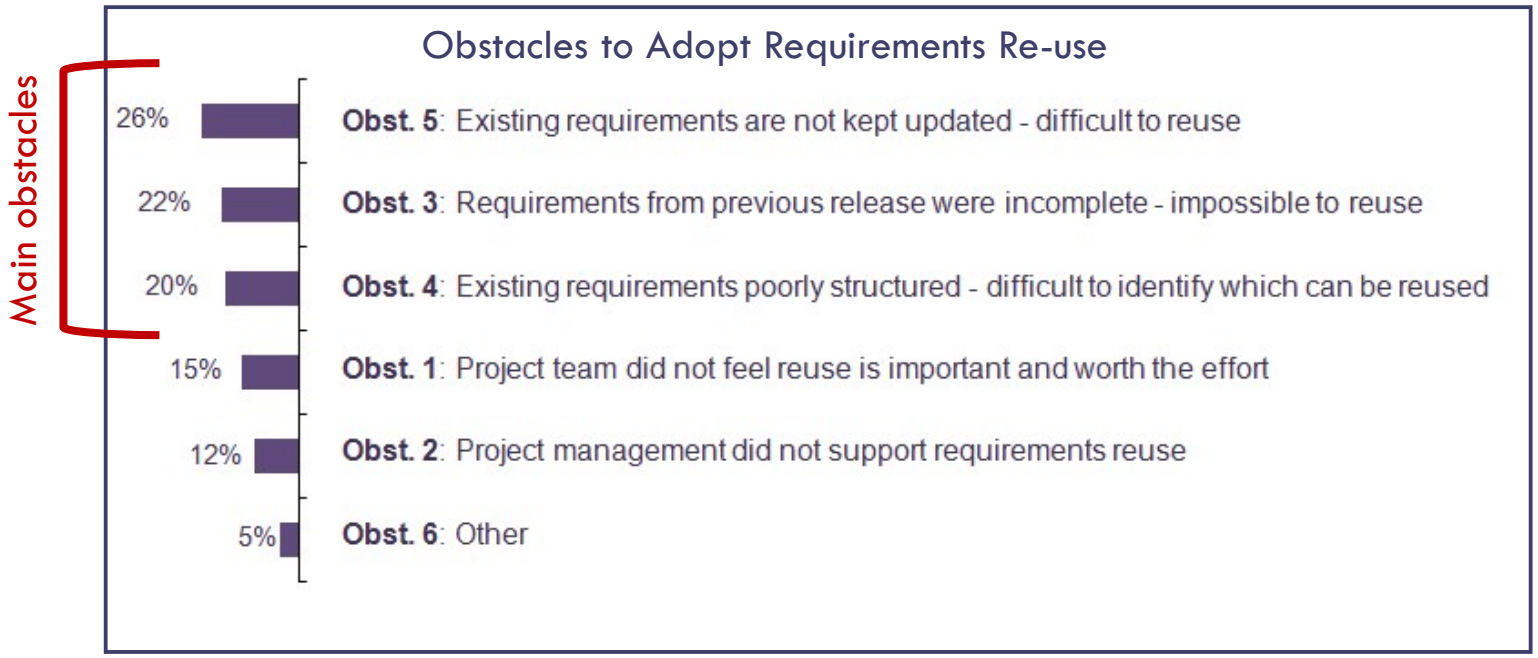
Benefits to Maintain and Reuse Requirements

- The term "requirements maintenance" means that requirements artifacts, initially produced in previous releases, are kept updated to be re-used in future releases.
- There are two main reasons to maintain requirements over time:
  - Perform impact analysis of software changes (most change requests, on average from 65% to 85%, overlap with the existing functionality),
  - Reduce the cost of requirements development.

Despite the benefits, requirements maintenance and reuse are not common practices in the IT industry.

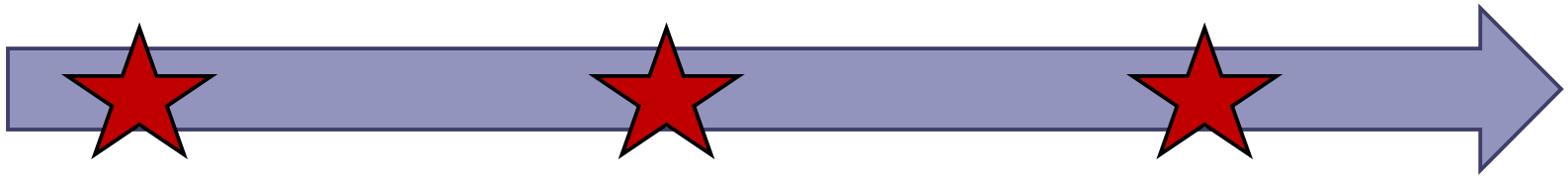
Requirements Reusability: The State of the Practice (2010 Survey Data)

According to the survey conducted in the Global IT community in 2010, only 59% of the survey participants re-used requirements on their projects.



The main obstacles to adopt requirements reuse all relate to poor structure and incompleteness of existing requirements.

History of the Aspect-Oriented Methodology



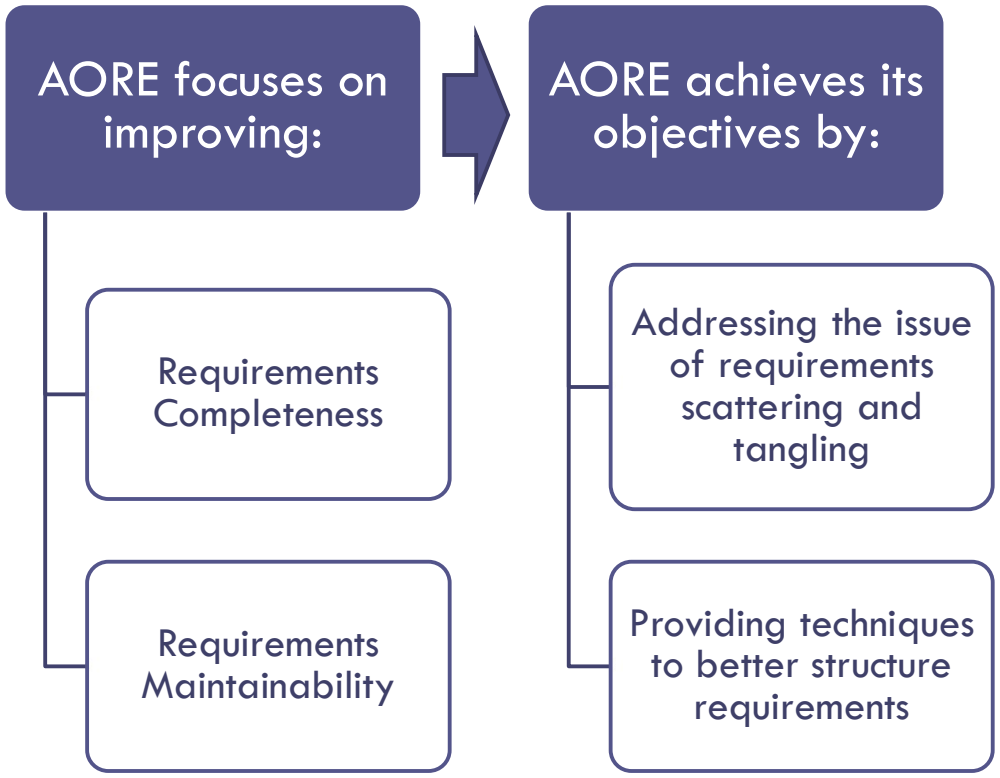
Gregor Kiczales introduced an aspect-oriented programming language.
The **Aspect-Oriented Software Development** field was born.

First articles on **Aspect-Oriented Requirements Engineering** (AORE) were published.

A few books and many research papers have been published.

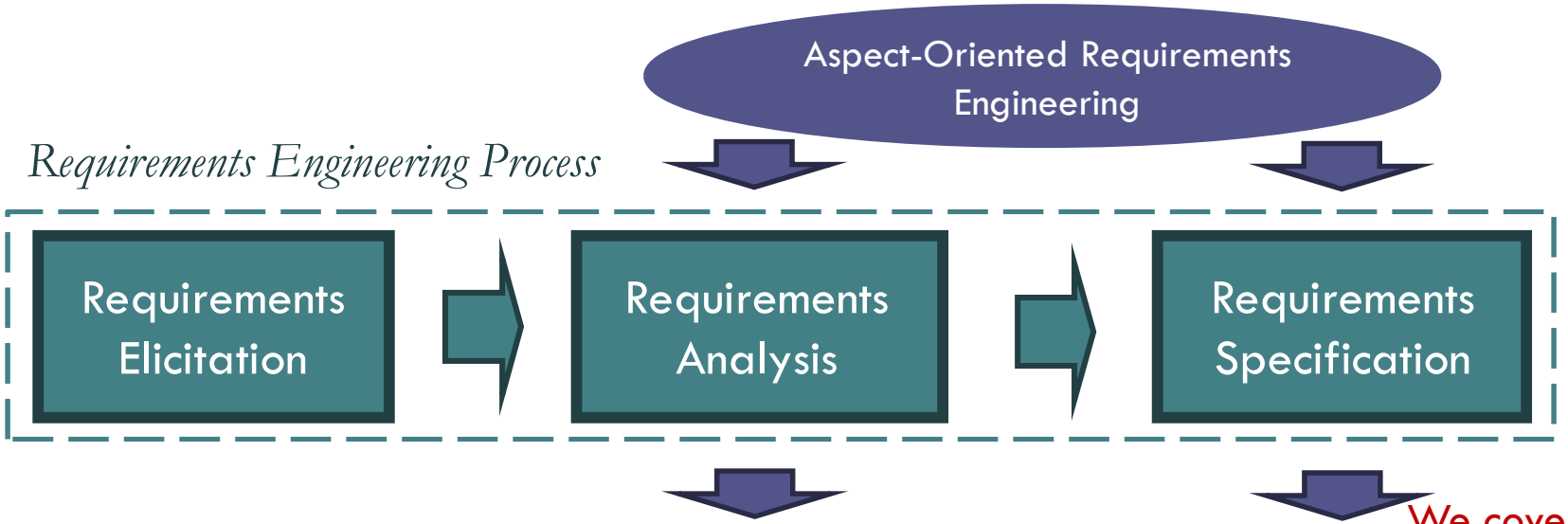
- AORE still remains little-known to most practitioners in the IT industry.
- The objective of this class is to raise awareness about AORE and its benefits.

AORE Methodology Objectives



AORE does not replace, but rather complements any of the conventional requirements methodologies.

Summary of the AORE Techniques



We cover next semester

The focus of this class

Techniques	▪ Separation of Concerns ▪ UML Modeling ▪ Aspect Interactions ▪ **Requirements Composition Table (RCT)**	▪ Join Point ▪ Pointcut ▪ Advice ▪ Composition Rules
Benefits	Better structure of requirements, improved change impact analysis and requirements maintainability	Improved requirements completeness

AORE vs. Other Methodologies

- AORE is not a replacement for any of the existing methodologies.
- AORE offers techniques for requirements analysis and specification that can be applied within any of the existing methodologies.
- AORE adoption on software projects has an easy entry point.

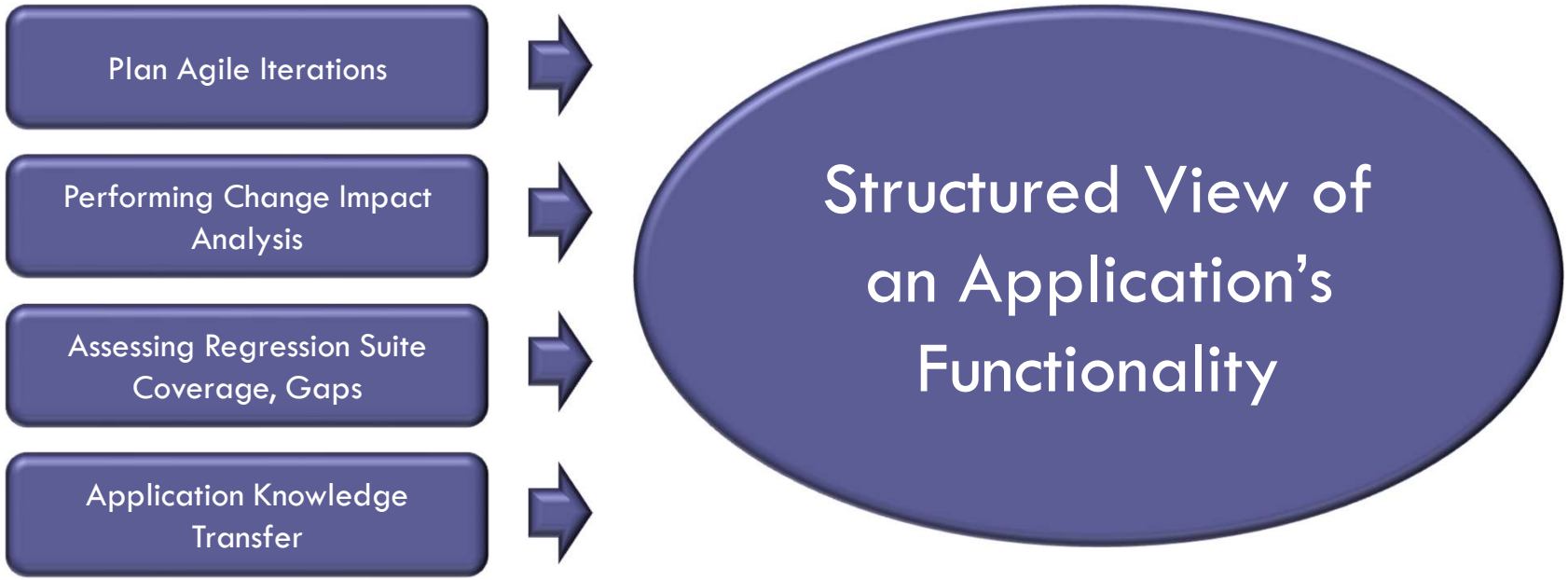
Part II.
Aspect-Oriented Requirements Analysis (RCT Technique)

References

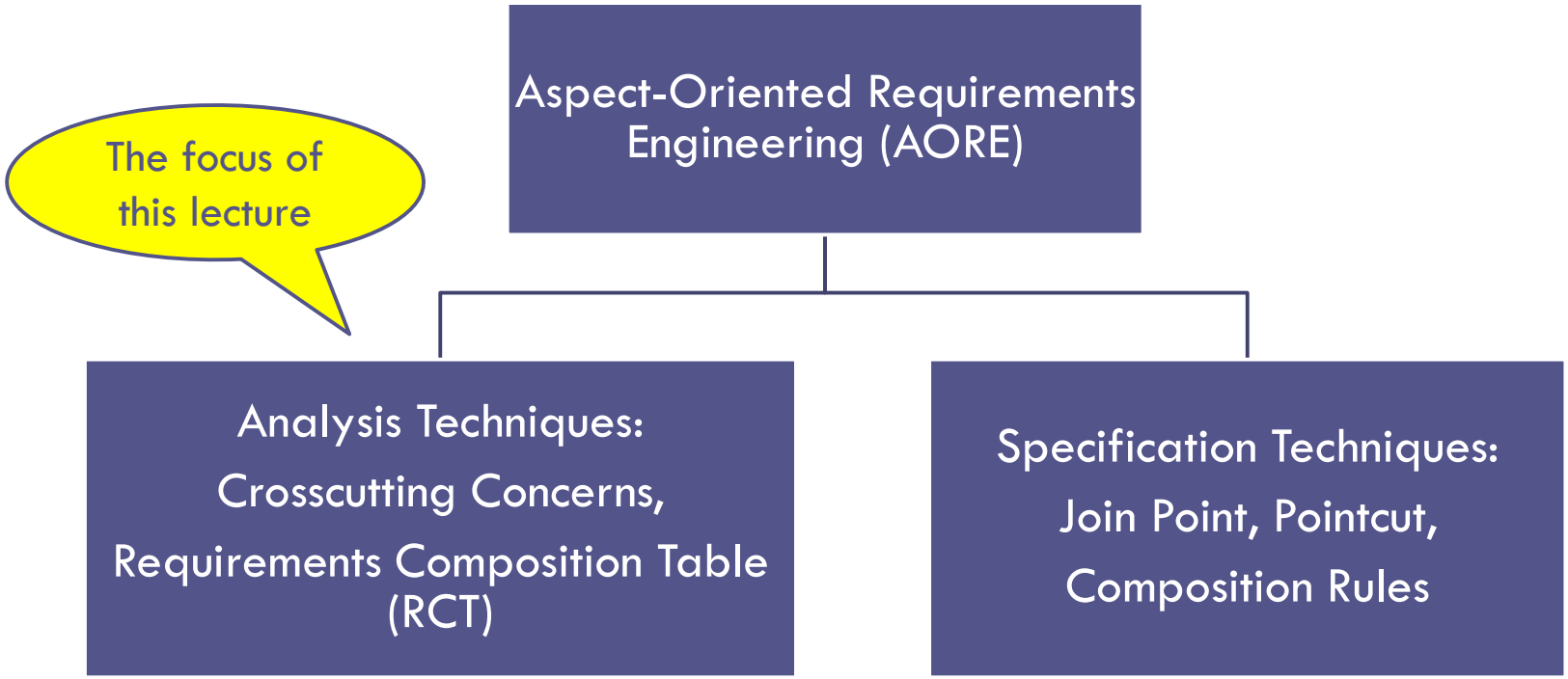
- Y. Chernak “Requirements Composition Table Explained”, 20th IEEE International Requirements Engineering Conference, 2012
- Y. Chernak “Dataflow Modeling with Crosscutting Concerns and a Concept Lattice”, IEEE Software, November 2014

A Need for a Holistic and Structured View

- Various software project tasks require a holistic and structured view of an application's functionality.
- Requirements Composition Table (RCT) has proven to be an effective solution.

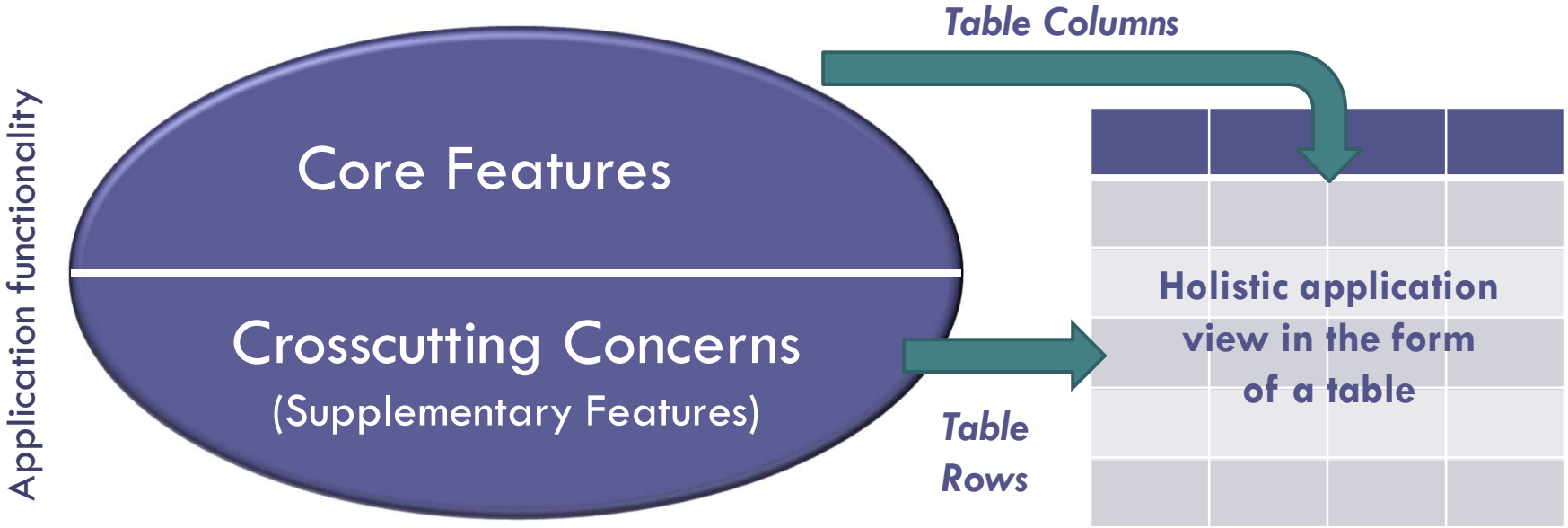


What is a Requirements Composition Table?



- RCT is a requirements analysis artifact introduced in AORE.
- RCT captures a complete inventory of an application's features, structured by **core** features and **crosscutting** concerns.

Two Perspectives of Aspect-Oriented Analysis



AORE applies the SoC principle and studies two categories of requirements:

Core features *capture basic application functionality and, when executed, produce a tangible and distinct business result.*

Supplementary features *do not produce business results by themselves, but rather complement core features.*

Most supplementary features are scattered across the application and tangled with core features; in AORE they are called *crosscutting concerns*.

RCT Example – Trading Application

- In AORE, we analyze the impact of crosscutting concerns on core features and capture results in the form of a Requirements Composition Table (RCT).
- RCT presents the application functionality as a binary relation between core features (table columns) and crosscutting concerns (table rows).
- RCT provides a common language (reference) for all parties on a project team.

Application XYZ: Requirements Composition Table

List of Concerns	01. Trade Processing																		
	01.01 New Instruction	01.02 Trade Correction	01.03 Repo Close	01.04 Trade Cancellation	01.05 Correction Filtering	01.06 Correct A	01.07 Correctio	01.08 Inflight T	01.09 Manual F	01.10 Finalize I	01.11 Repo Off	01.12 Hide Errc	01.13 Missing I	01.14 Accrued Interest Batch	01.15 Override Errors	01.16 Swift BIC Code, Account Resolution	01.17 Summarize Trade Counts	01.18 Missing Accrued Interest Repair	01.19 Rebalance Trade File
Core Functionality	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	0	1	1	0	1	1	0	1	0	0	1	0	1	1	0
Crosscuting Concerns																			
ET-In - Internal Entitleme	1	1	1	1	0	1	1	0	1	1	0	1	0	0	1	0	0	1	0
ET-Ex - External Entitler	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
STY - Security	1	1	1	1	0	1	0	0	1	1	0	0	0	0	1	0	0	0	0
CS - Client Setup	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0	1
PT - Product Type	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	1	1
CST - Cash Status	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TST - Trade Status	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1
FXST - FX Status	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
TPST - Template Status	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FV - Field Validation	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0
DDV - Data-Dependency	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
DDD - Data-Driven Defau	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
CL - Calculations	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1
ER - Enrichment	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1
MP - Mapping	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
CC - Concurrency	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
CN - Connectivity	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
RG - Region	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1
DF-In - Data Flow In	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	0
DF-Out - Data Flow Out	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	1	0
SI-In System Interface (i	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0
SI-Out - System Interfac	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EML-In - Email In	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EML-Out - Email Out	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
ADT-In - Internal User Au	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ADT-Ex - External User	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CA - Cache	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
FS - File Status	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
ExH - Exception Handlin	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1
PF - Performance	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Inventory of Core Features

Crosscutting Concerns

Composition of Concerns (binary relation)

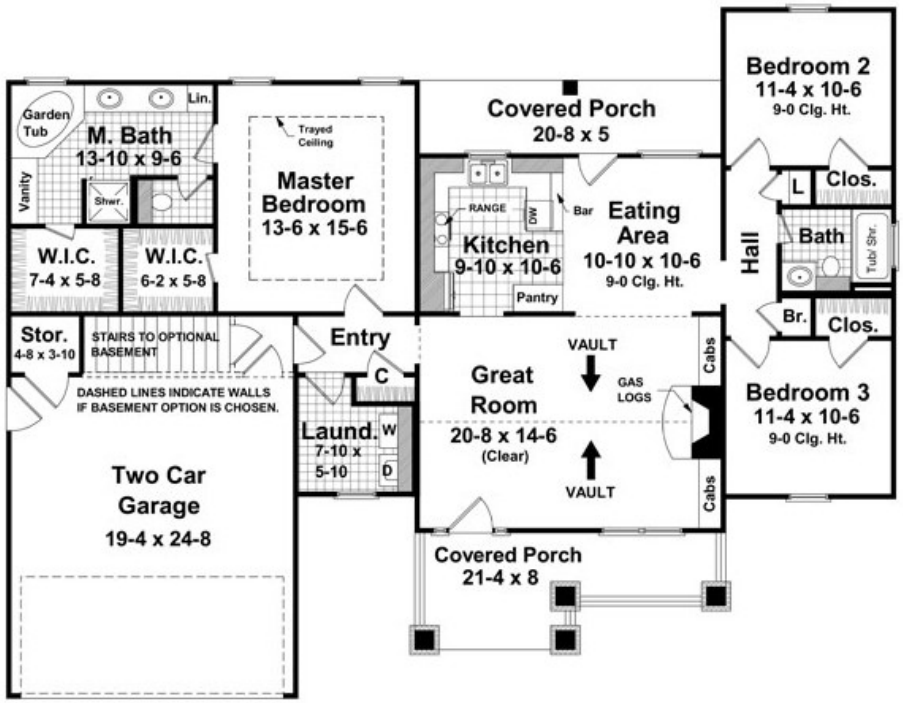
1 – means a core feature is impacted by a crosscutting concern

0 – means no impact

RCT as an "Application's Floor Plan"

Architect

Electrician



Carpenter

Plumber

- All parties on a construction team need the same frame of reference, i.e., a ***Floor Plan.***
- Similarly, all parties on an application team need the same frame of reference, which can be a Requirements Composition Table.

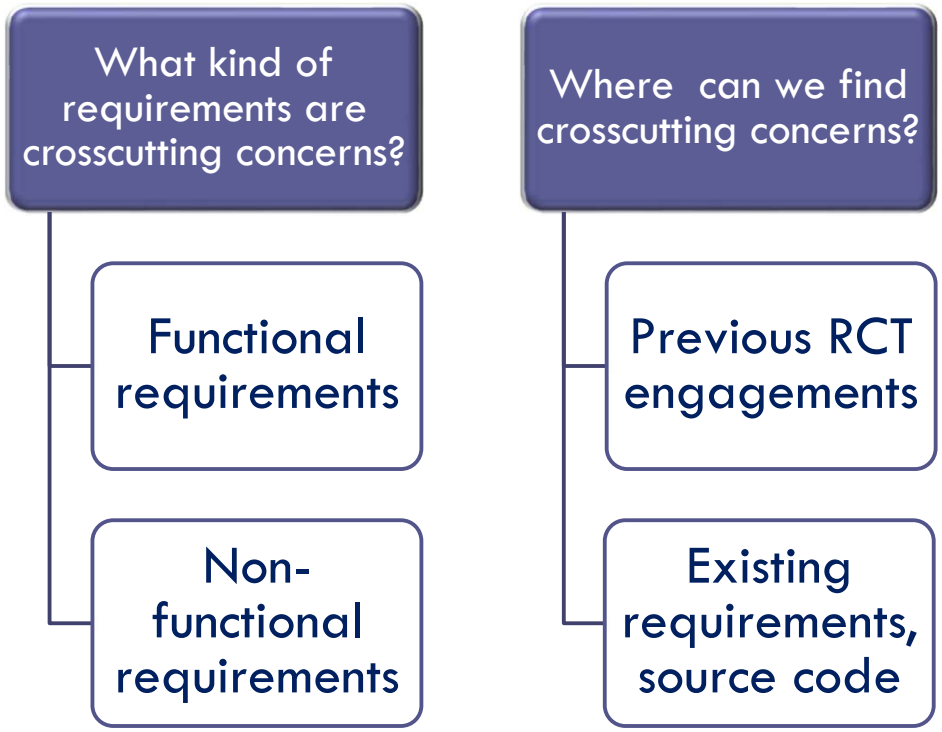
Characteristics of Crosscutting Concerns

Supplementary features that we can analyze and model as *crosscutting concerns* should comply with the following characteristics:

1. They cannot be invoked directly by end-users; to be executed, they need the context of a core feature.
2. When invoked, they can impact a core feature's context by constraining, interrupting, or enhancing the core feature flow.
3. They are sufficiently scattered, i.e., they should impact at least three or four core features.

In practice, many categories of crosscutting concerns are generic and repeat across various applications in the same business domain.

Identifying Crosscutting Concerns



When we develop a new RCT, a good starting point would be to review a list of crosscuts from previous RCT engagements.

Most Common Crosscuts and Their Meaning

ID	Crosscut Description
ET - Entitlements	This concern relates to different user access privileges (roles) and how they impact the behavior of core features.
FV - Field Validation	This concern relates to validating individual data-entry fields.
DDV - Data-dependency Validation	This concern relates to validating a combination of values of two or more fields.
DDD - Data-driven defaults	This concern relates to populating field default values based on another field value.
DF-In, DF-Out	Transaction dataflow In/Out among an application's core features.
SI-In, SI-Out	Transaction dataflow In/Out between an application and upstream and downstream systems.
CL - Calculations	This concern represents various "behind-the-screen" calculations that are executed in the context of core features.
CC – Concurrency Control	This concern relates to simultaneous data manipulation by more than one user. For example, both users see the same order. User 1 tries to modify this order, whereas User 2 tries to execute this order.
CN - Connectivity Validation	This concern relates to the architecture of an application where the front-end can be disconnected from the back end and that would change the behavior of some core features.

See Appendix C for a more complete list of Crosscuts.

Most Common Crosscuts and Their Impact on Core Features

ID	Crosscut Description	Impact Type
ET	User entitlements	Constrain
FV	Data-entry field validation	Interrupt flow
DDV	Data-dependency validation	Interrupt flow
DDD	Data-driven defaults	Add detail
DF-In DF-Out	Transaction dataflow In/Out among an application's core features	Add detail
SI-In SI-Out	Transaction dataflow In/Out between an application and its upstream and downstream systems	Add detail
CC	Concurrency control for multi-user systems	Interrupt flow
CN	Front-end connectivity validation	Interrupt flow
PF	Application performance	Add detail

Examples of Crosscutting Concerns

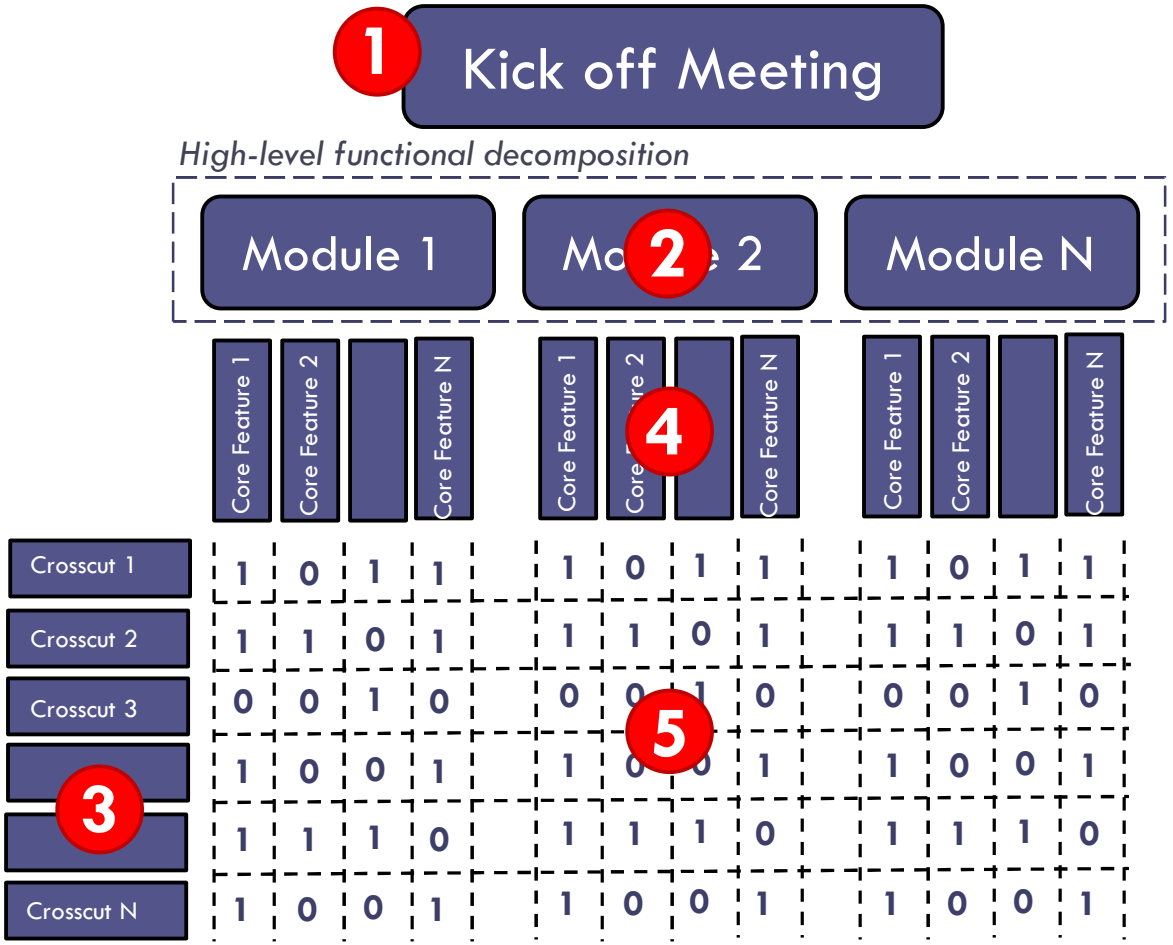
Categories of Crosscutting Concerns for Investment Banking Applications			
Access & Entitlements	**Client Setup & System Configuration**	**Product Type & Transaction Status**	**Data Validation & Manipulation**
ET-In - Internal Entitlements	CS - Client Setup	PT - Product Type	FV - Field Validation
ET-Ex - External Entitlements	SC - System Configuration	CLT - Collateral Type	DDV - Data-Dependency Validation
AUT - Authorization	RG - Region	TRC - Trade Category	DDD - Data-Driven Defaults
STY - Security	LE - Legal Entity	TIF - Time-In-Force	DER - Data Enrichment
		OST - Order Status	DMP - Data Mapping
Audits & Alerts	**Data & Transaction Flow**	TRST - Trade Status	CL - Calculations
ADT-In - Internal User Audit	MB - Message Broadcasting	CST - Cash Status	CC - Concurrency
ADT-Ex - External User Audit	CN - Connectivity	SST - Settlement Status	
ALR - Alerts	SDF - Static Data Flow	FXST - FX Status	
ExH - Exception Handling	TDF-In - Transaction Data Flow In	TPST - Template Status	
	TDF-Out - Transaction Data Flow Out	FST - File Status	
Non-functional Concerns	SI-In - System Interface In	DST - Deal Status	
CA - Cache	SI-Out - System Interface Out	TST - Transaction Status	
PF - Performance	NOT - Notifications	AST - Agreement Status	

- Commonly, a practical number of crosscutting concerns for a given application can be from 20 to 40 items.
- This number primarily depends on the application complexity and level of abstraction to identify crosscutting concerns.
- Crosscutting concerns can represent both functional and non-functional categories of requirements.

Steps to Produce an RCT

1. Conduct a kick off meeting, agree on the RCT engagement mission, identify SMEs and establish their commitments
2. Identify application modules (a.k.a. functional areas), assign SMEs
3. Identify crosscutting concerns, agree on their meaning
4. For each module, identify the inventory of core features
5. Analyze the impact of crosscuts; compose each core feature with related crosscuts (0/1)
6. Validate and refine a draft RCT

W5 Deliverable



Naming and Enumerating Core Features

- Identify core features with a numeric, composite, unique ID [ModuleID.FeatureID]. Example "02.01 Create User Account"
- Name a core feature with a verb-noun phrase that states the user's goal.
- Use concrete, "strong" verbs instead of generalized, weaker ones. Weak verbs may indicate uncertainty.
    - Strong Verbs: create, merge, calculate, migrate, receive, archive, register, activate.
    - Weaker Verbs: make, report, use, copy, organize, record, find, process, maintain, list.
- Do not combine two different goals in the same core feature. Bad example - "create and modify account".

Composition of Concerns Example: Hotel Management Application

Use Case: 01.01 Create Guest Reservation

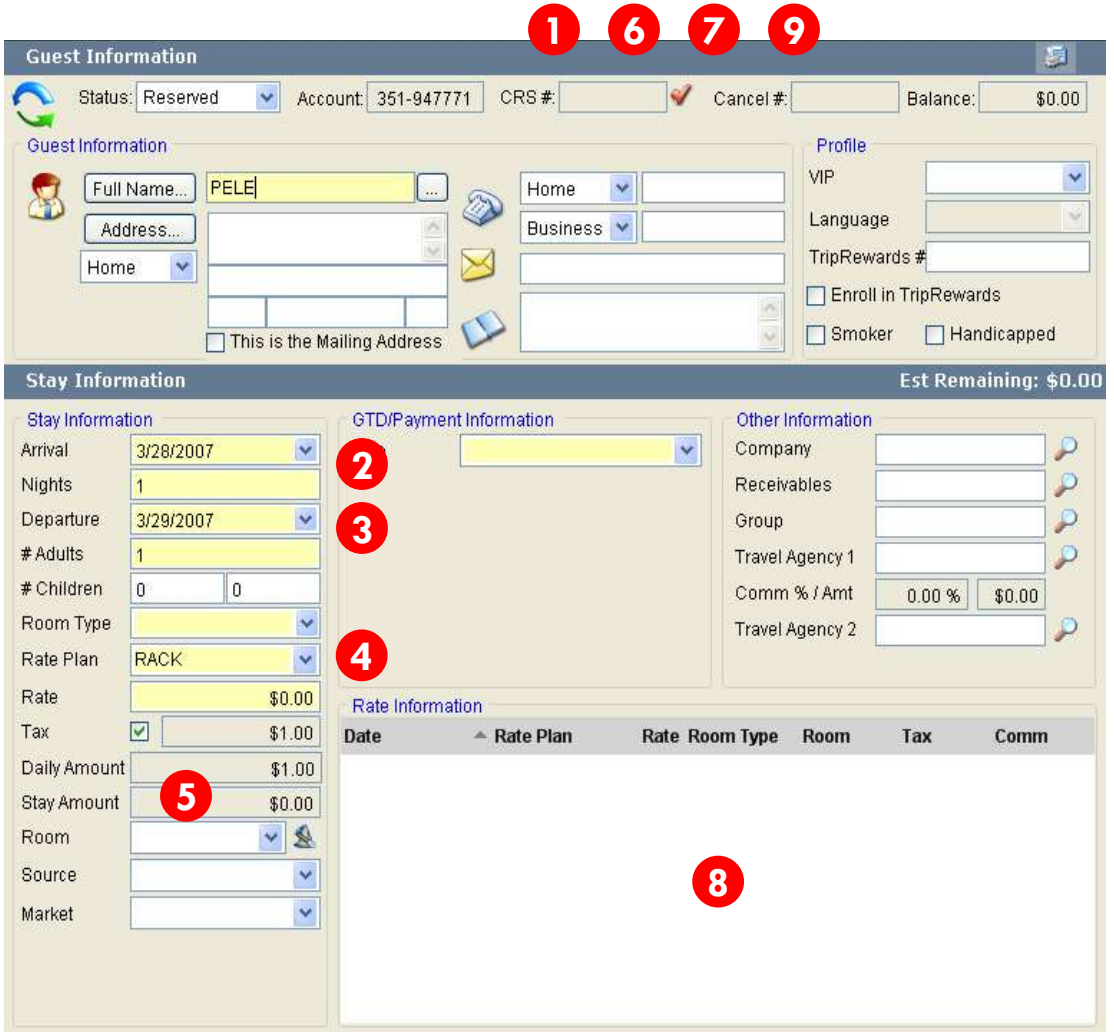
Use Case Scenario

1. The use case begins when the Front Desk Clerk intends to create a new guest reservation (or check-in a walk-in guest).
2. User enters a guest's personal, stay, and payment information.
3. System provides the Rate&Plan information and available room inventory based on the stay information.
4. User selects the room type and rate plan.
5. System calculates the stay amount.
6. User submits the reservation.
7. System creates a new reservation and sends the local reservation to the central reservation system.

A core feature specification is not complete without capturing the impact of crosscutting concerns.

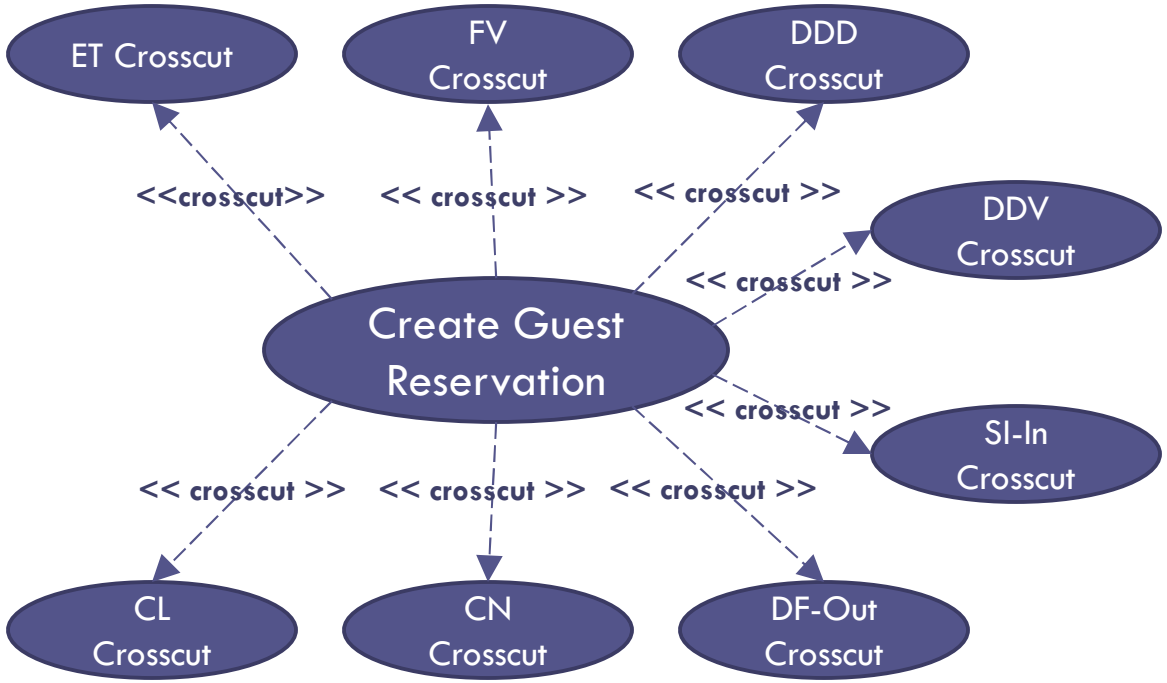
Class Exercise: Composition of Concerns

	RCT Fragment Concern Types	Create Guest Reservation
	Core Functionality	1
	GUI - User Interface	1
	Crosscutting Concerns	
1	ET - Entitlements	1
2	FV - Field Validation	1
3	DDV - Data-Dependency Validation	1
4	DDD - Data-Driven Defaults	1
5	CL - Calculations	1
	CC - Concurrency	0
6	CN - Connectivity	1
	TST - Transaction Status	0
	DF-In - Data Flow In	0
7	DF-Out - Data Flow Out	1
8	SI-In - System Interface In	1
9	SI-Out - System Interface Out	1



Example: Crosscut Composition Modeling with a UML Diagram

RCT Fragment Concern Types	Create Guest Reservation
Core Functionality	1
GUI - User Interface	1
Crosscutting Concerns	
ET - Entitlements	1
FV - Field Validation	1
DDV - Data-Dependency Validation	1
DDD - Data-Driven Defaults	1
CL - Calculations	1
CC - Concurrency	0
CN - Connectivity	1
TST - Transaction Status	0
DF-In - Data Flow In	0
DF-Out - Data Flow Out	1
SI-In - System Interface In	1
SI-Out - System Interface Out	1



- At this time, no agreed and standard UML notation exists for modeling crosscutting concerns.
- I. Jacobson proposed that crosscuts be modeled as extension use cases.
- Crosscuts do not fully comply with both <<include>> and <<extend>> relationships, a new type is needed.

Join Points Explained

- To complete a core feature specification with related crosscuts, we need to mark the core feature's flow where a crosscut behavior is inserted.
- We can do that by using Join Points [JP].

Definition

- Join Points are points in the core feature flow where a crosscut's behavior is combined with the core feature.
- A crosscutting concern can be inserted into a core feature flow either **before, after,** or **around** its respective join point.

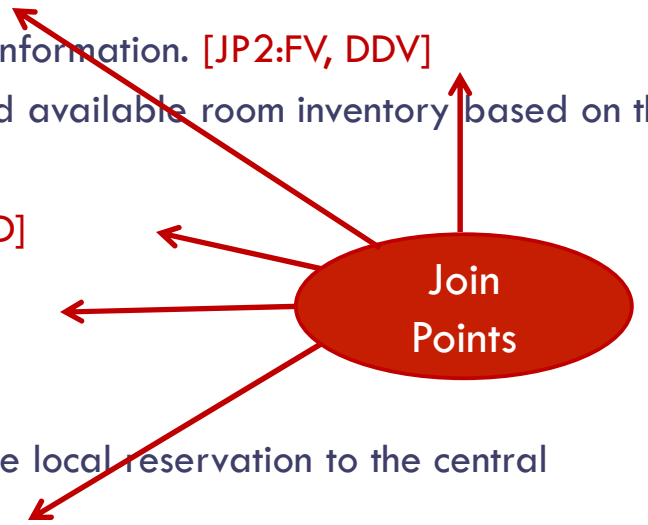
A given core feature specification is not complete without indicating how it is impacted by related crosscuts.

Use Case Completed with Crosscuts

Use Case: 01.01 Create Guest Reservation

Use Case Description

1. The use case begins when the Front Desk Clerk intends to create a new guest reservation (or check-in a walk-in guest). [JP1:ET]
2. User enters a guest's personal, stay and payment information. [JP2:FV, DDV]
3. The system provides the Rate&Plan information and available room inventory based on the stay information. [JP3:SI-In]
4. User selects the room type and rate plan. [JP4:DDD]
5. The system displays the stay amount. [JP5:CL]
6. User submits the reservation.
7. The system creates a new reservation and sends the local reservation to the central reservation system. [JP6:CN, DF-Out, SI-Out]



By using Join Points we can produce a more complete core feature specification .

Summary of the RCT Benefits

- RCT is a **requirements analysis** artifact that captures a **structured** and **holistic** view of the application functionality in the form of a **binary relation.**
- RCT has proven to be one of the most valuable project artifacts as it can effectively support the following tasks:
  - **Iterative and Incremental Development** (*see W8 Lecture*)
  - Requirements Reverse Engineering
  - **Software Change Impact Analysis** (*discussed in Part III*)
  - Test Coverage Assessments
  - Functional and Regression Testing
  - Application Knowledge Transfer

RCT Ownership and Maintenance

Why Maintain an RCT?

- An RCT is intended to accurately represent the existing application functionality that can evolve from release to release.
- This makes it necessary to maintain the RCT to ensure its consistency with the application functionality.

Assigning an RCT Owner

- A project team needs to discuss who will own the RCT. This could be a Lead BA, a Lead QA Analyst, or a Technical Lead.
- The RCT Owner will be responsible for maintaining an RCT, in particular, for updating the RCT based on the changes implemented in the last release.

RCT Engagements on Wall Street

- For the last ten years, the RCT technique has been implemented for over a dozen Wall Street projects in investment banking.
- There were three categories of sponsors of these engagements:
  - **Developers,** who needed to improve change impact analysis and better plan new releases.
  - **Testers,** who needed to assess test coverage and identify gaps in their existing regression test suites.
  - **Business Analysts**, hired for renovation projects, who needed a holistic view of the legacy system to be replaced with the new application.
- Although these engagements had different objectives, each needed a holistic view of an application's functionality and structure. Using an RCT was an effective solution.

RCT References

1. A. Moreira, R. Chitchyan, J. Araújo, A. Rashid, *Aspect-Oriented Requirements Engineering*, Springer-Verlag, 2013, ISBN 978-3-642-38639-8 (Chapter 15)
2. A. Rashid, A. Moreira, and J. Araújo, "Modularization and Composition of Aspectual Requirements", *Proceedings of 2nd International Conference on Aspect-Oriented Software Development (AOSD)*, 2003, ACM, pp. 11-20.
3. Y. Chernak "Requirements Composition Table Explained", in proceedings of the 20th IEEE International Requirements Engineering Conference, Chicago, IL, September 2012, pp. 273-278
4. Y. Chernak "Dataflow Modeling with Crosscutting Concerns and a Concept Lattice", *IEEE Software, November 2014, pp. 70-78*
5. Y. Chernak "Building a Foundation for Structured Requirements. Part 1", *Better Software, January 2009*, pp. 90-96
6. Y. Chernak "Building a Foundation for Structured Requirements. Part 2", *Better Software, March 2009*, pp. 40-46
7. Y. Chernak "Mind the Gap: Using a Requirements Composition Table to Assess Test Coverage", *Better Software, March 2008*, pp.38-44

Part III.
RCT-based
Change Impact Analysis

Reference

Moreira, R. Chitchyan, J. Araújo, A. Rashid, *Aspect-Oriented Requirements Engineering*, Springer-Verlag, 2013

Chapter 15: Y. Chernak "Implementing Aspect-Oriented Requirements Analysis for Investment Banking Applications"

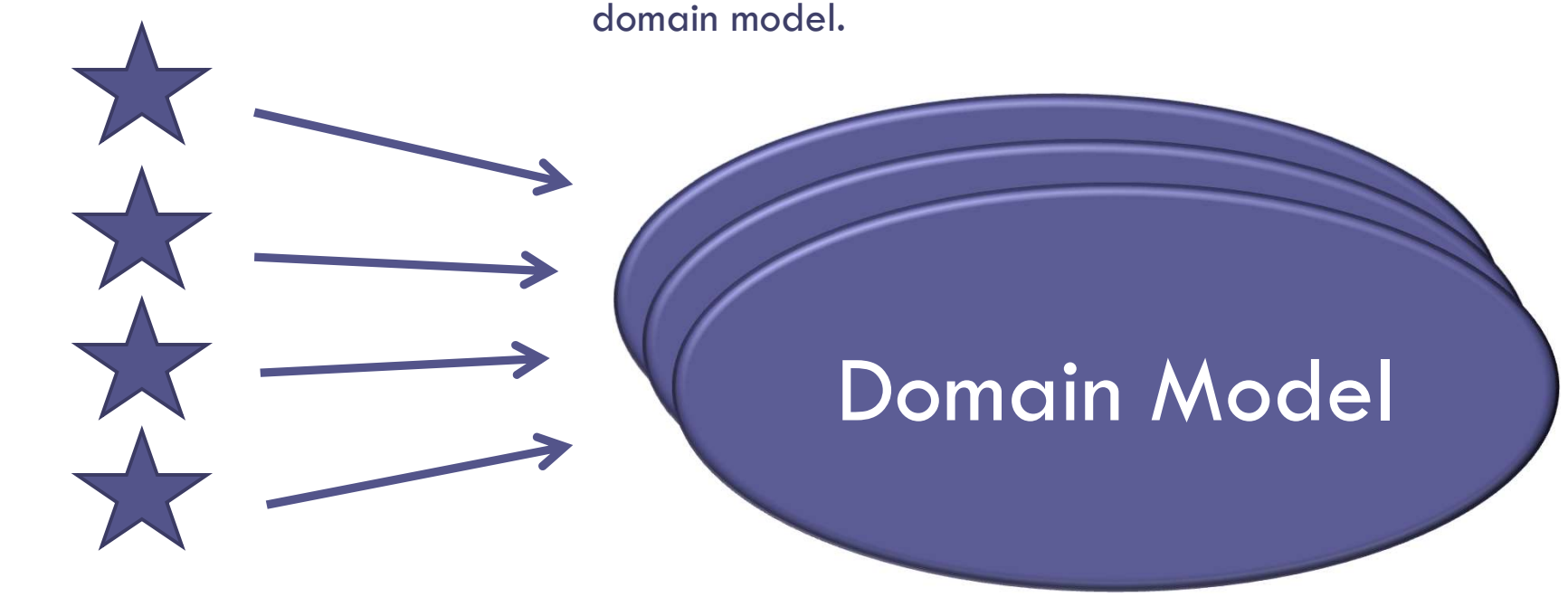
Change Impact Analysis – Why?

- Business applications are always evolving driven by constantly changing business conditions.
- Most release changes (on average from 65% to 85% per release) overlap with the existing application functionality.
- A lack of understanding of the impact of changes is a common reason for production instability and incidents, which can cause significant financial losses to the business.

Change impact analysis is a critical task of software maintenance projects and it has been the focus of research for decades.

Change Impact Analysis Concept

- To effectively analyze the impact of changes, we need a domain model and use it as a frame of reference to perform impact analysis and capture CIA results.
- When CIA is performed from the perspectives of different stakeholders, we need more than one domain model.



A Common Misconception

CR ID	Change Request Description
CR 29802	implement Options Regulatory Fee initiated by the CBOE
CR 30578	implement optimistic concurrency control for trade processing
CR 31897	implement multiple trade cancellation
CR 32182	upgrade WebLogic Server to a new version

Wrong!

To produce functional requirements, all we need to do is just to add details to business/user requirements (change requests).

Functional Specification Document (FSD) for a release

RCT-based Change Impact Analysis

In practice, most change requests overlap with the existing application features.

- CIA objective is to "translate" high-level change requests into low-level product features.
- We use an RCT as a frame of reference to analyze the impact of each change request (CR), one at a time, to identify the impacted product features.

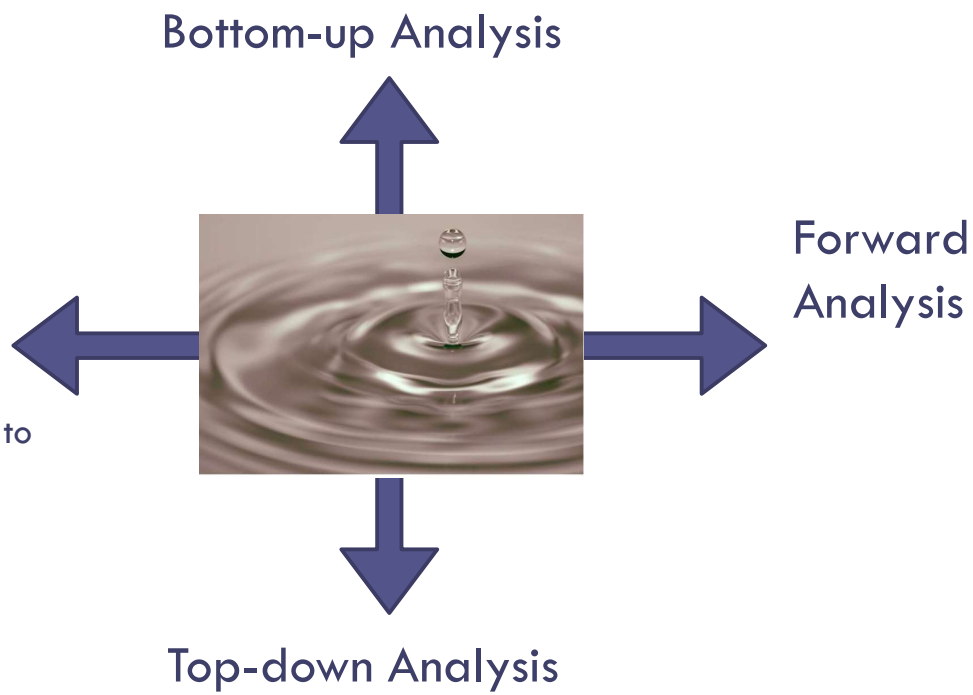
CR ID	Change Request Description
CR 29802	implement Options Regulatory Fee initiated by the CBOE
CR 30578	implement optimistic concurrency control for trade processing
CR 31897	implement multiple trade cancellation
CR 32182	upgrade WebLogic Server to a new version

Application XYZ: Requirements Composition Table

List of Concerns	01. Trade Processing																		
	01.01 New Instruction	01.02 Trade Correction	01.03 Repo Close	01.04 Trade Cancelation	01.05 Correction Filtering	01.06 Correct Allocation - External	01.07 Correction Allocation - Flip Action	01.08 Inflight Trade Processing	01.09 Manual Price Input	01.10 Finalize Pricing	01.11 Repo Offleg Publishing	01.12 Hide Errors	01.13 Missing Products Repair	01.14 Accrued Interest Batch	01.15 Override Errors	01.16 Swift BIC Code, Account Resolution	01.17 Summarize Trade Counts	01.18 Missing Accrued Interest Repair	01.19 Rebalance Trade File
Core Functionality	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	0	1	1	0	1	1	0	1	0	0	1	0	1	1	0
Crosscuting Concerns																			
ET-In - Internal Entitlements	1	1	1	1	0	1	1	0	1	1	0	1	0	0	1	0	0	1	0
ET-Ex - External Entitlements	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
STY - Security	1	1	1	1	0	1	0	0	1	1	0	0	0	0	1	0	0	0	0
CS - Client Setup	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0	1
PT - Product Type	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	1	1
CST - Cash Status	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TST - Trade Status	1	1	1	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	1
FXST - FX Status	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TPST - Template Status	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FV - Field Validation	1	1	1	1	1	1	1	1	0	0	0	1	1	0	1	0	0	1	0
DDV - Data-Dependency Validation	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DDD - Data-Driven Defaults	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CL - Calculations	1	1	1	1	1	1	1	1	0	0	0	1	1	0	1	0	0	1	1
ER - Enrichment	1	1	1	1	1	1	1	1	0	0	0	1	1	0	1	0	0	1	1
MP - Mapping	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CC - Concurrency	1	1	1	1	1	1	1	1	0	0	0	1	0	0	1	0	0	0	0
CN - Connectivity	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
RG - Region	1	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	1
DF-In - Data Flow In	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0
DF-Out - Data Flow Out	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	0
SI-In System Interface (inbound)	1	1	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0
SI-Out - System Interface (outbound)	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
EML-In - Email In	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EML-Out - Email Out	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0
ADT-In - Internal User Audit	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
ADT-Ex - External User Audit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CA - Cache	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
FS - File Status	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
ExH - Exception Handling	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1
PF - Performance	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

RCT Benefits: Four CIA Directions

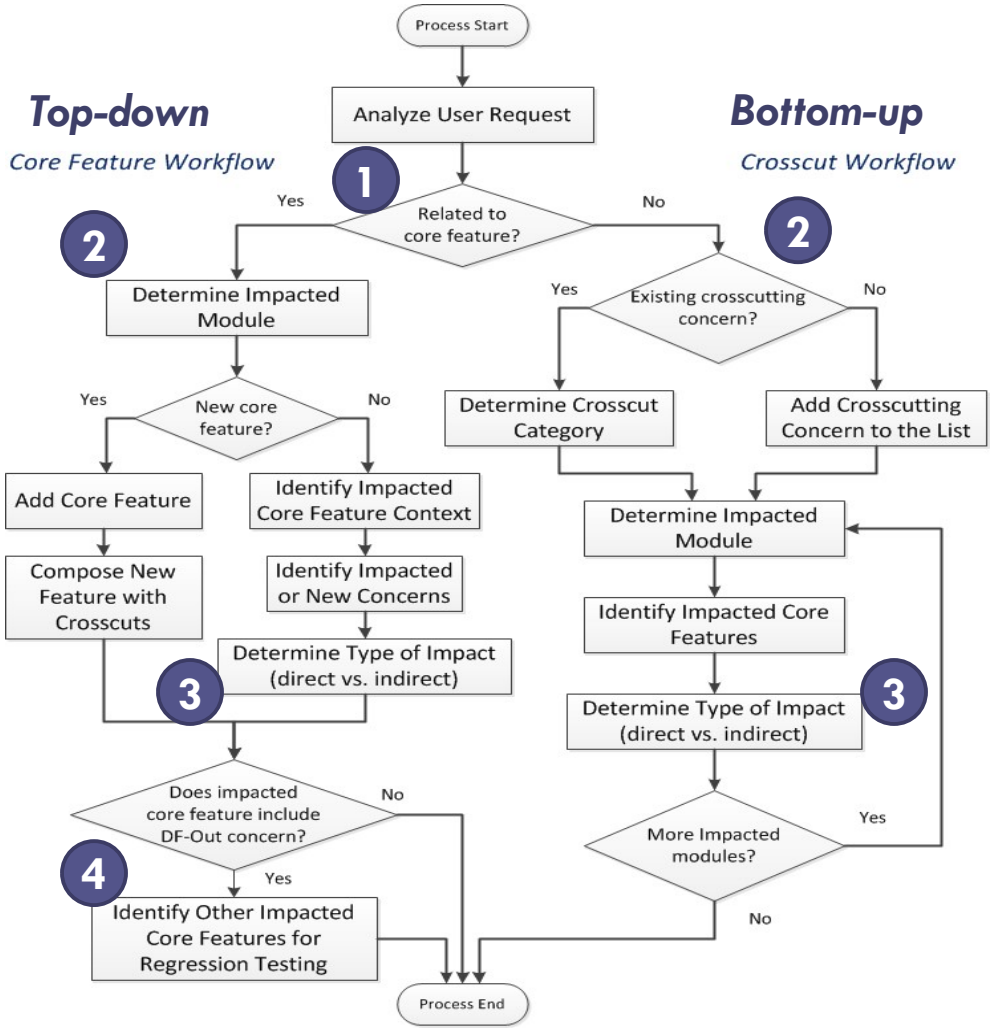
- **Reference Model**: a reference model should represent a holistic view of the application functionality and capture dependencies among the application's features.
- **Use of RCT**: using an RCT as a reference model has proven effective to perform this type of analysis.
- **RCT Benefits**: the benefit of using the RCT is that it allows us to investigate the impact of changes in all four directions:



- **Top-down** analysis – from core features down to crosscuts.
- **Bottom-up** analysis – from crosscuts up to core features.
- **Backward, Forward** analysis – based on data flows In/Out.

Change Impact Analysis Workflow

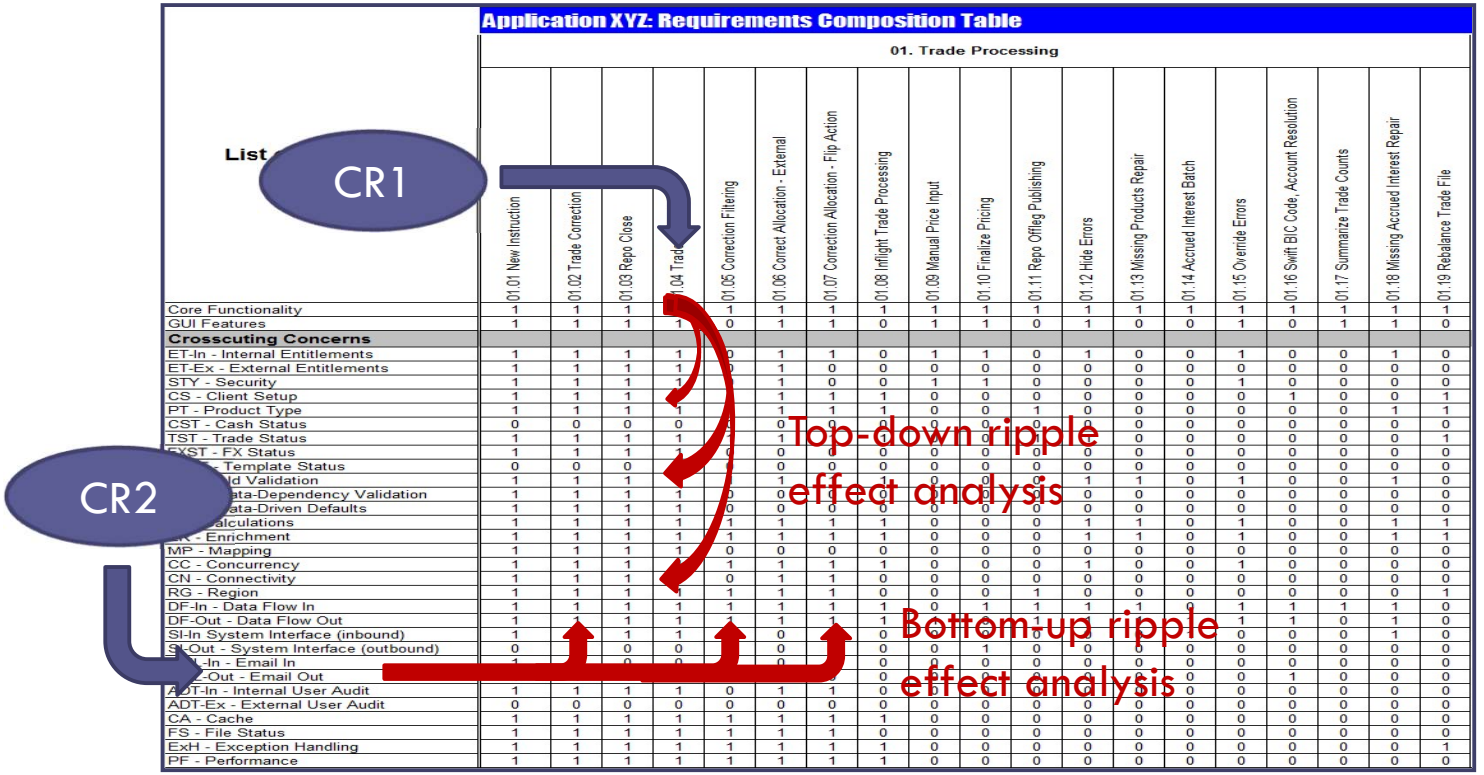
1. The procedure starts with the question "Does a change request relate to a core feature or a crosscutting concern?"
2. Depending on the answer, we follow one or another branch in the CIA workflow (see flow-chart).
3. For impacted features, we determine the type of impact, i.e., ***direct*** or ***indirect*** impact.
4. We investigate the backward and forward ripple effects via data flows.



CIA: Top-Down vs. Bottom-Up Analysis

A change to one feature can present risks to other features:

- CR1: changes to core features propagate down to crosscutting concerns (top-down analysis).
- CR2: changes to crosscutting concerns propagate up to core features (bottom-up analysis).



Application XYZ: Requirements Composition Table

List	01. Trade Processing																		
	01.01 New Instruction	01.02 Trade Correction	01.03 Repo Close	01.04 Trade	01.05 Correction Filtering	01.06 Correct Allocation - External	01.07 Correction Allocation - Flip Action	01.08 Inflight Trade Processing	01.09 Manual Price Input	01.10 Finalize Pricing	01.11 Repo Offleg Publishing	01.12 Hide Errors	01.13 Missing Products Repair	01.14 Accrued Interest Batch	01.15 Override Errors	01.16 Swift BIC Code, Account Resolution	01.17 Summarize Trade Counts	01.18 Missing Accrued Interest Repair	01.19 Rebalance Trade File
Core Functionality	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	0	1	1	0	1	1	0	1	0	0	1	0	1	1	0
Crosscuting Concerns																			
ET-In - Internal Entitlements	1	1	1	1	0	1	1	0	1	1	0	1	0	0	1	0	0	1	0
ET-Ex - External Entitlements	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
STY - Security	1	1	1	1	1	1	0	0	1	1	0	0	0	0	1	0	0	0	0
CS - Client Setup	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0	1
PT - Product Type	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	1	1
CST - Cash Status	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TST - Trade Status	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1
FXST - FX Status	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[illegible] Template Status	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[illegible]d Validation	1	1	1	1	1	1	1	1	0	0	0	1	1	0	1	0	0	1	0
[illegible]ata-Dependency Validation	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[illegible]ata-Driven Defaults	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[illegible]alculations	1	1	1	1	1	1	1	1	0	0	0	1	1	0	1	0	0	1	1
[illegible] - Enrichment	1	1	1	1	1	1	1	1	0	0	0	1	1	0	1	0	0	1	1
MP - Mapping	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CC - Concurrency	1	1	1	1	1	1	1	1	0	0	0	1	0	0	1	0	0	0	0
CN - Connectivity	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
RG - Region	1	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	1
DF-In - Data Flow In	1	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	1	1	0
DF-Out - Data Flow Out	1	1	1	1	1	1	1	1	0	1	1	1	0	0	1	1	0	1	0
SI-In System Interface (inbound)	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
SI-Out - System Interface (outbound)	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
[illegible]-In - Email In	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[illegible]-Out - Email Out	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0
ADT-In - Internal User Audit	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
ADT-Ex - External User Audit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CA - Cache	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
FS - File Status	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
ExH - Exception Handling	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1
PF - Performance	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

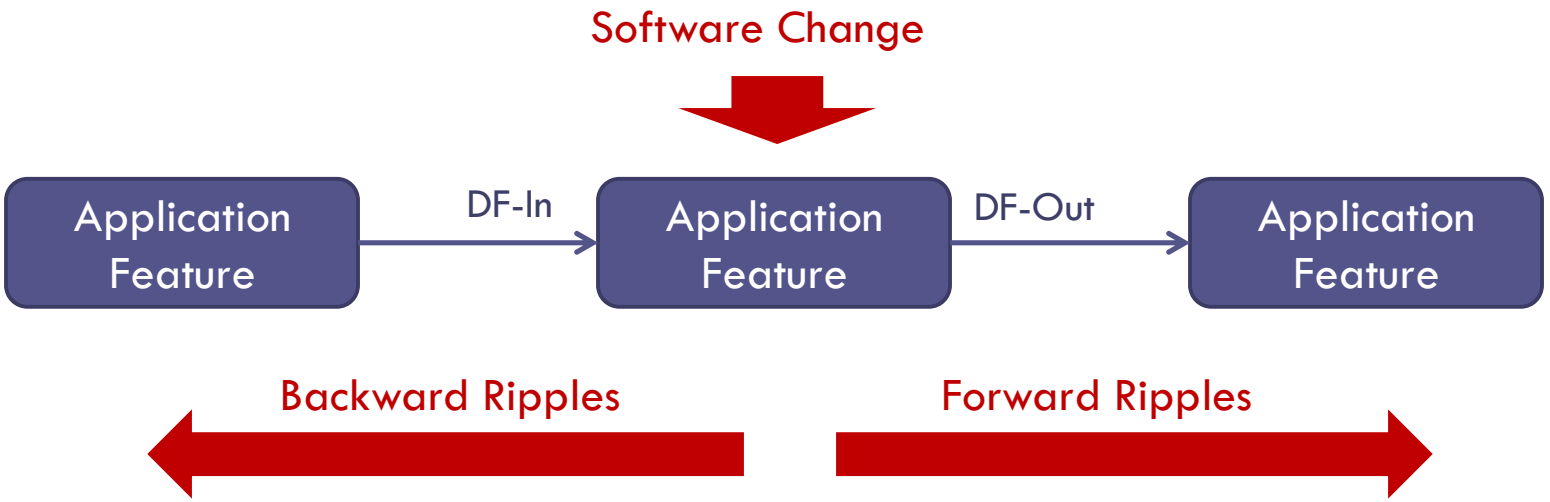
CIA: Backward vs. Forward Analysis

Backward Ripple Effects

- The investigation of backward ripples is intended to answer the question "*how does the data get here?*"

Forward Ripple Effects

- The investigation of forward ripples is intended to answer the question "*where is the data used from here?*"



Impact Type: Direct vs. Indirect Types

Direct Impact

- The Direct type of impact means that to implement a requested change we need to change the application code.
- In this case, impacted features should be qualified as ***directly*** impacted and included in the scope of functional testing.

Indirect Impact

- The Indirect type of impact means that we do not change the application code of a given feature.
- However, we still have some quality concerns and this feature should be qualified as ***indirectly*** impacted and included in the scope of regression testing.

In an RCT, we use the impact indicators, i.e., the letters (D)irect and (I)ndirect, to mark the type of impact.

Capturing CIA Results in RCT

CR 31897 overlaps with the existing core feature 01.04

CR 29802 requires adding a new core feature 01.11

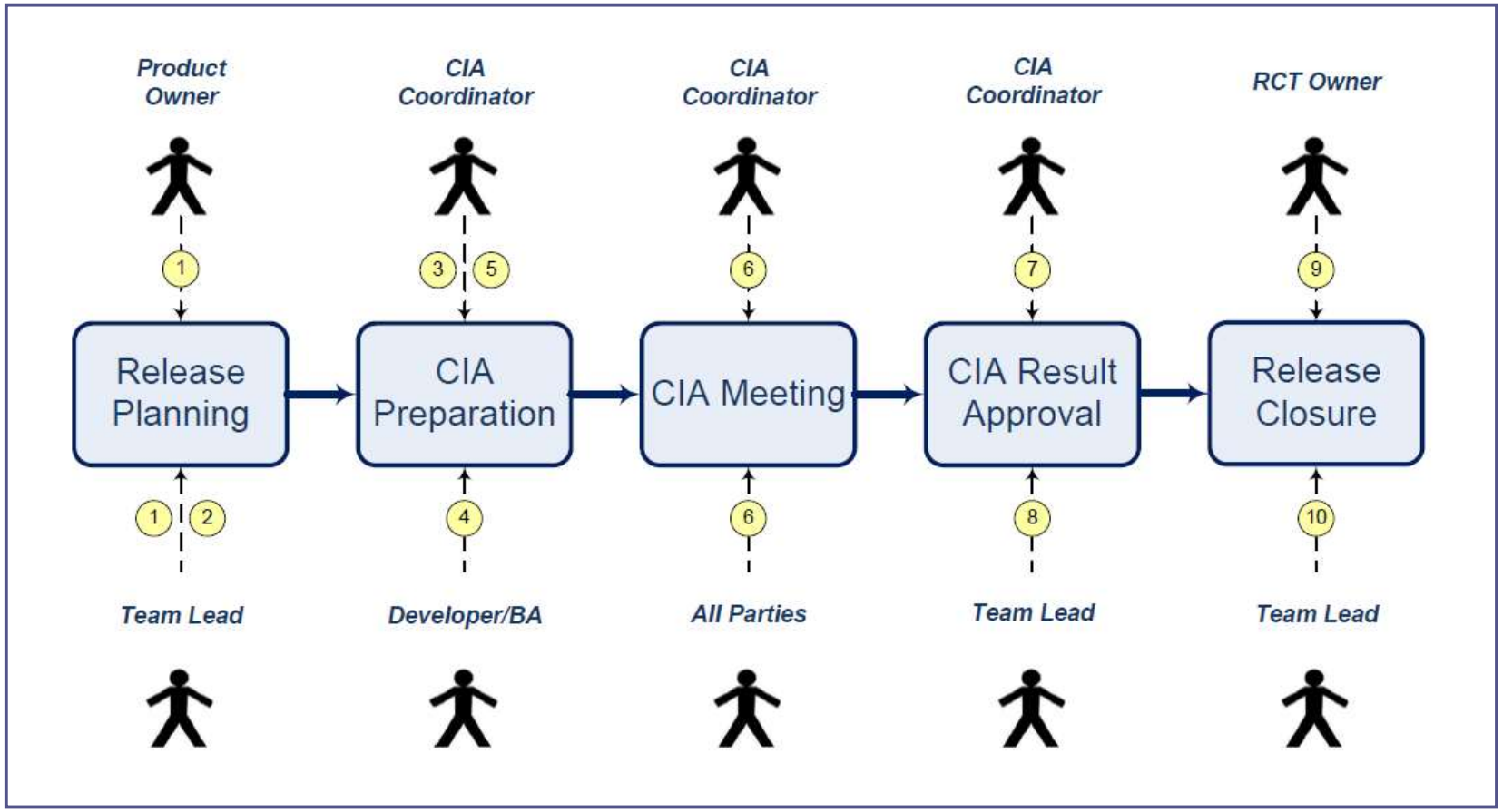
	01.01 Ne	01.02 Tra	01.03 Repo Close	01.04 Trade Cancellation	01.05 Manual Price Input	01.06 Fir	01.07 Re	01.08 Ac	01.09 Ac	01.10 Accrued Interest Repair	01.11 ORF Management
Core Functionality	29802(D) 30578(D)	29802(D) 30578(D)	30578(D)	29802(D) 30578(D) 31897(I)	1	1	31897(I)	1	31897(I)	31897(I)	29802(D)
GUI Features	1	1	1	31897(D)	1	1	0	0	0	1	29802(D)
Crosscutting Concerns											
ET-In - Internal Entitlements	1	1	1	31897(I)	1	1	0	0	0	1	29802(D)
ET-Ex - External Entitlements	1	1	1	1	0	0	0	0	0	0	0
STY - Security	1	1	1	1	1	1	0	0	0	0	0
CS - Client Setup	1	1	1	1	0	0	0	0	1	0	29802(D)
PT - Product Type	1	1	1	31897(I)	0	0	1	0	0	1	29802(D)
CST - Cash Status	0	0	0	0	0	0	0	0	0	0	0
TST - Trade Status	1	1	1	31897(D)	0	0	1	0	0	0	0
FXST - FX Status	1	1	1	1	0	0	0	0	0	0	0
TPST - Template Status	0	0	0	0	0	0	0	0	0	0	0
FS - File Status	1	1	1	1	0	0	0	0	0	0	0
FV - Field Validation	1	1	1	31897(D)	0	0	0	0	0	1	29802(D)
DDV - Data-Dependency Validation	1	1	1	1	0	0	0	0	0	0	0
DDD - Data-Driven Defaults	1	1	1	31897(D)	0				0	0	29802(D)
CL - Calculations	1	1	1	1	0				0	1	29802(D)
ER - Enrichment	1	1	1	1	0				0	1	0
MP - Mapping	1	1	1	1	0				0	0	0
CC - Concurrency	30578(D)	30578(D)	30578(D)	30578(D) 31897(D)	0				0	0	29802(D)
CN - Connectivity	1	1	1	31897(D)	0	0	0	0	0	0	29802(D)
RG - Region	1	1	1	31897(I)	0	0	1	0	0	0	0
DF-In - Data Flow In	29802(D)	29802(D)	1	29802(D)	0	1	31897(I)	0	31897(I)	31897(I)	0
DF-Out - Data Flow Out	1	1	1	31897(D)	1	0	1	1	1	1	29802(D)
SI-In - System Interface In	1	1	1	1	0	0	1	0	0	1	0
SI-Out - System Interface Out	29802(D)	29802(D)	1	31897(I) 29802(D)	0				0	0	0
EML-In - Email In	1	0	0	0	0				0	0	0
EML-Out - Email Out	1	1	1	1	0				1	0	0
ADT-In - Internal User Audit	1	1	1	31897(I)	0				0	0	29802(D)
ADT-Ex - External User Audit	0	0	0	0	0	0	0	0	0	0	0
CA - Cache	1	1	1	1	0	0	0	0	0	0	0
ExH - Exception Handling	1	1	1	31897(I)	0	0	0	0	0	0	0
PF - Performance	32182(I)	32182(I)	0	31897(I) 32182(I)	0	0	0	0	0	0	0

(D) – indicates direct impact

(I) – indicates indirect impact

CIA Process Phases and Roles

The CIA process involves multiple project roles.



See task descriptions on the next slide.

CIA Process Roles and Responsibilities

1. **Product Owner** and **Team Lead**: finalize the release scope, i.e., a list of change requests (a.k.a. business requirements).
2. **Team Lead**: assigns change requests to developers to analyze and implement.
3. **CIA Coordinator**: schedules a CIA meeting.
4. **Assigned Developers/BA**: prepare a CIA case for each change request.
5. **CIA Coordinator**: reconciles inputs from Developers, prepares the meeting materials.
6. **CIA Coordinator**: conducts a CIA meeting (Product Owner, Developers, QA parties are invited):
   a. **Developers/BA**: present a CIA case for each change request.
   b. **All Parties**: examine and validate the CIA results.
   c. **CIA Coordinator**: facilitates a discussion, takes notes.
7. **CIA Coordinator**: publishes the final CIA results after the meetings.
8. **Team Lead**: approves the CIA results for the release.
9. **RCT Owner**: at the end of a release, updates the RCT and produces a new version to be approved and baselined.
10. **Team Lead**: approves the latest RCT version.

To make the CIA process more practical, some roles can be combined and assigned to the same team member.

Conducting a CIA Meeting

CIA meeting parties

CIA should be performed as a formal [1 hr] session where all release parties are invited – the product owner, the developers assigned to implement a release, and the QA personnel assigned to test the release.

CIA meeting outcome

The CIA meeting outcome is the agreement among all release parties on the scope of release implementation and testing:

- Directly impacted features represent the scope of **functional testing**.
- Indirectly impacted features represent the scope of **regression testing**.
- Impacted system interface concerns indicate the scope of **end-to-end testing**.

Part I: Key Points

- Various application features can be tangled with and impact each other. This presents challenges to develop **complete** and **maintainable** specifications of software requirements.
- AORE methodology provides techniques to develop better-structured requirements that can help us improve requirements completeness, maintainability, and reduce the cost of software development.
- AORE methodology does not replace, but rather complements any of the existing requirements methodologies.
- AORE methodology emerged fifteen years ago, but still remains little-known to most IT practitioners.

Part II: Key Points

- AORE introduced the concept of crosscutting concerns. We discussed examples and the characteristics of crosscutting concerns.
- RCT is a requirements analysis technique introduced within the aspect-oriented methodology.
- RCT is used to capture a holistic and structured view of an application's functionality in the form of a binary relation.
- We discussed the steps to produce an RCT.
- The RCT technique can support various project tasks and can provide multiple benefits.
- We discussed RCT's ownership and maintenance.

Part III: Key Points

- In practice, most requested changes allocated to a new release, overlap with an application's existing functionality and require investigating the impact of changes.
- An RCT-based change impact analysis can be conducted in four directions.
- The types of impact of changes can be classified as Direct and Indirect.
- The *Direct* impact can help us better define the scope of functional testing.
- The *Indirect* impact can help us better justify the scope of regression testing.
- We discussed the CIA process – phases and roles.
- The CIA meeting should be planned as a formal session where all parties are invited; the meeting outcome is an agreement among the parties on the scope of release implementation and testing.

Exercises

Part I.

1. Explain what is the focus of aspect-oriented requirements engineering (AORE).
2. Explain the principle of "Separation of Concerns" and how it is applied in AORE.
3. Explain how AORE fits other requirements methodologies.
4. What is a crosscutting concern? How is it different from a core feature?

Part II.

1. Explain why we need a holistic and structured view of an application's functionality.
2. What are the two perspectives of the aspect-oriented analysis?
3. What is a Requirements Composition Table (RCT)?
4. Explain the characteristics of crosscutting concerns.
5. Give some examples of common crosscutting concerns.
6. What are the steps to produce an RCT?
7. What are the RCT benefits, what tasks can it support?

Part III.

1. Explain why an RCT can help us better perform change impact analysis.
2. What are the four directions to investigate the impact of changes using the RCT?
3. What is the difference between the direct vs. indirect types of impact and why it is important to understand?

Appendix A.

Crosscut Analysis Examples

Example 1: HVAC Office Suite, PO Entry

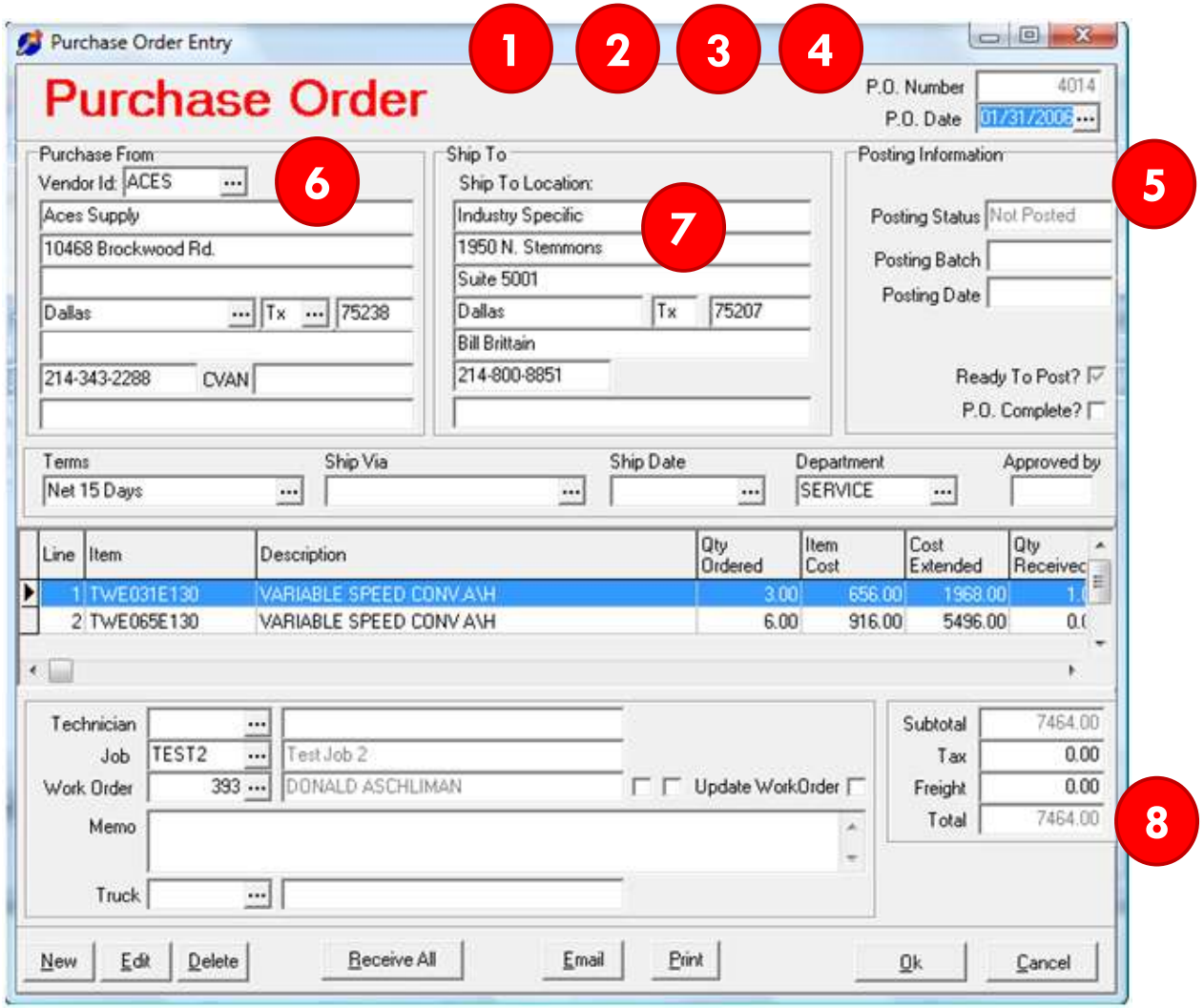
The HVAC Office suite is designed for service contractors.

Core Features

- New PO
- Edit PO
- Delete PO

Crosscuts

1. ET – Entitlements
2. CN – Connectivity
3. CC – Concurrency
4. DF-Out – Dataflow Out
5. PST – PO Status
6. DDD – Data-Driven Defaults
7. FV – Field Validation
8. CL – Calculations



Example 2: Warehouse Management System

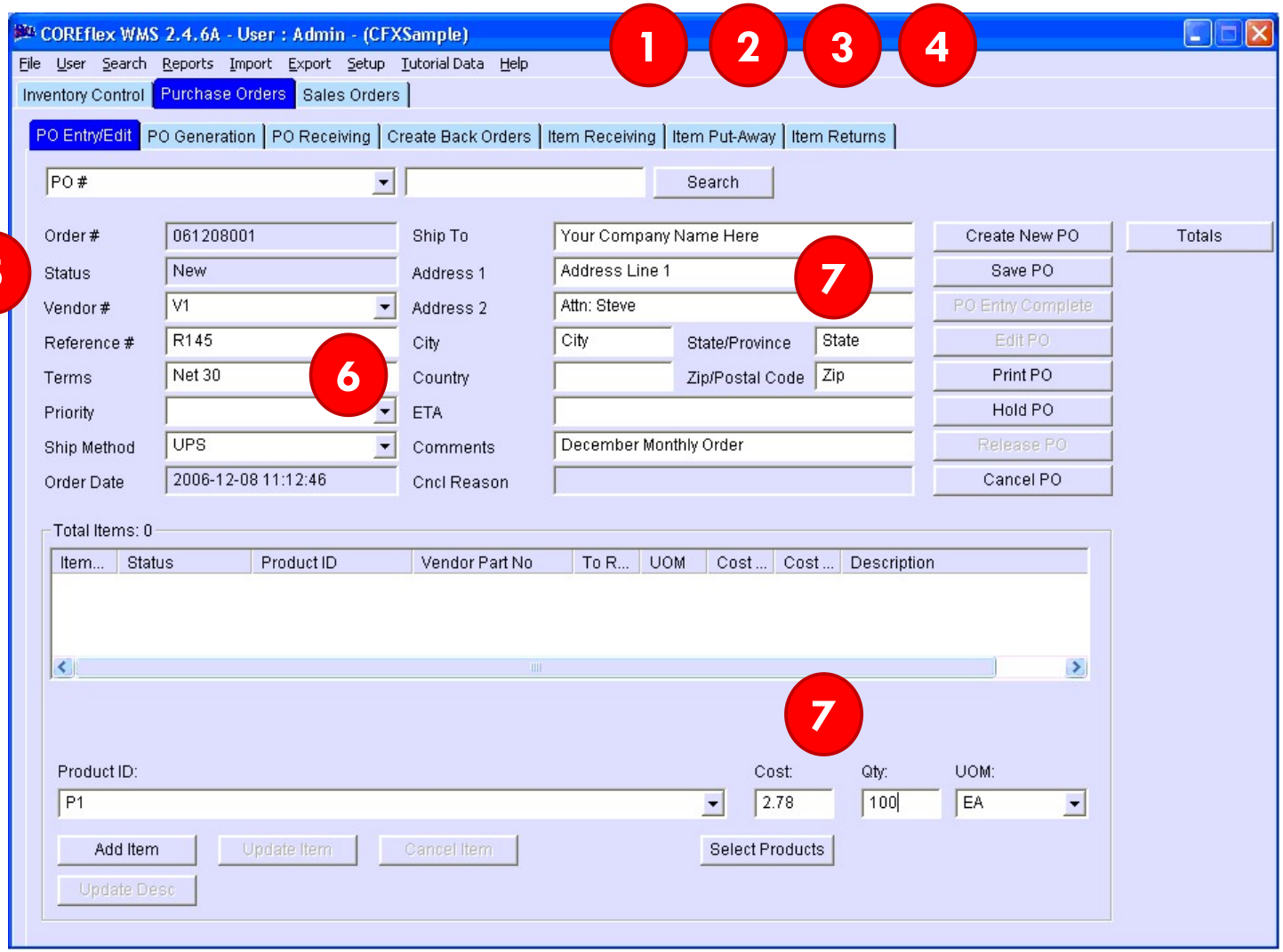
PO Entry/Edit Screen

<u>Core Features</u>

- New PO
- Edit PO
- Cancel PO

<u>Crosscuts</u>

1. ET – Entitlements
2. CN – Connectivity
3. CC – Concurrency
4. DF-Out – Dataflow Out
5. PST – PO Status
6. DDD – Data-Driven Defaults
7. FV – Field Validation



Example 3: Inventory Reporting System

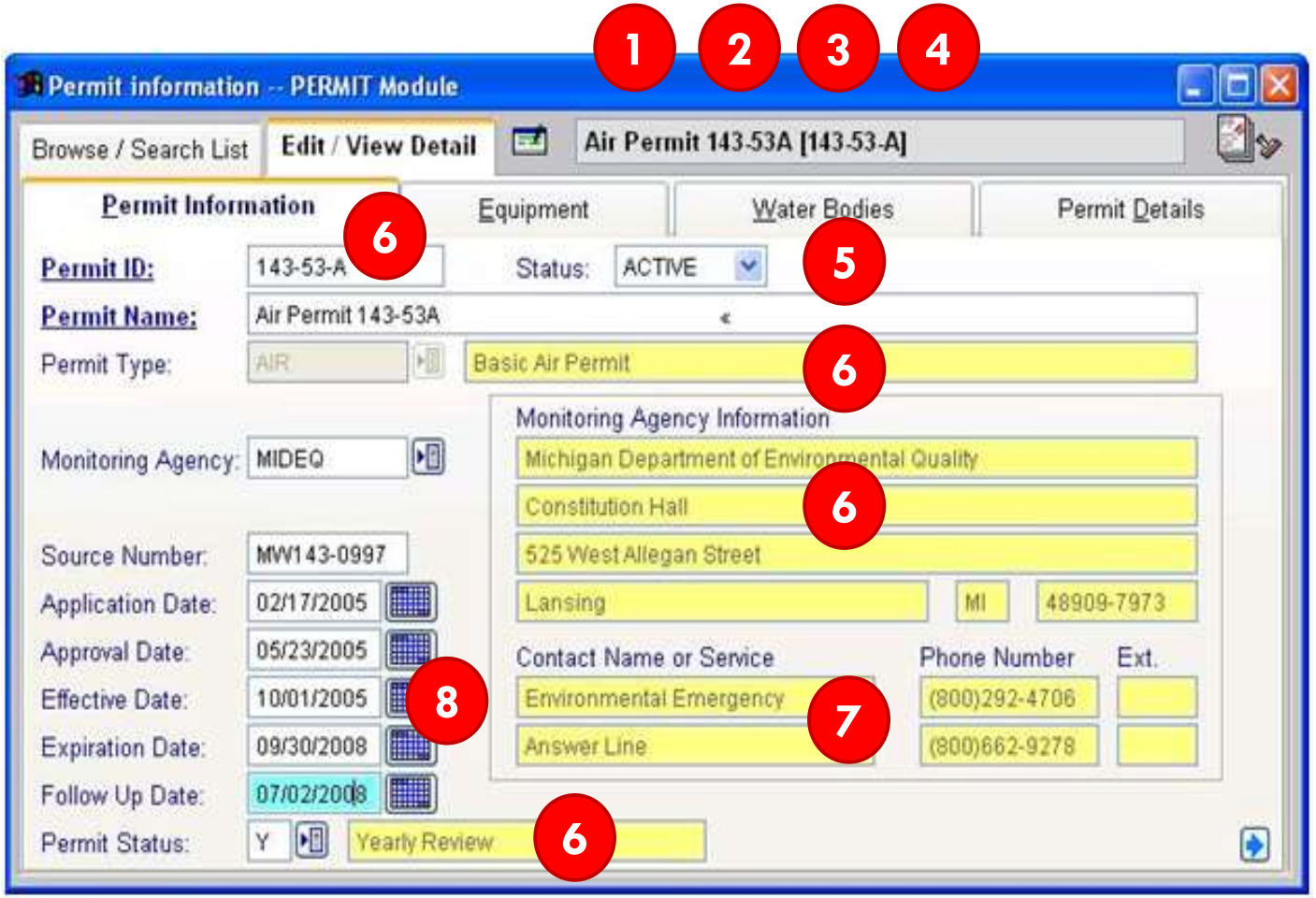
Permit Entry/Edit Screen

Core Features

- New Permit
- Edit Permit
- Cancel Permit

Crosscuts

1. ET – Entitlements
2. CN – Connectivity
3. CC – Concurrency
4. DF-Out – Dataflow Out
5. PST – Permit Status
6. DDD – Data-Driven Defaults
7. FV – Field Validation
8. DDV – Data Dependency Validation



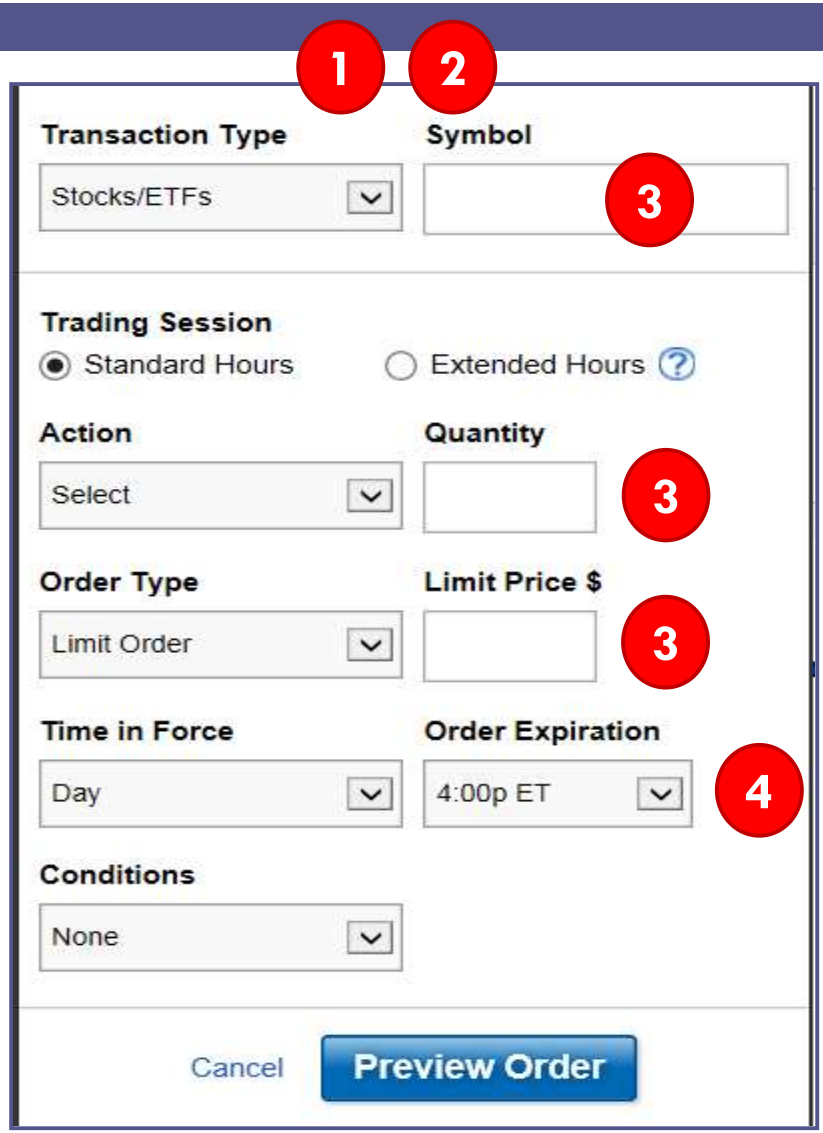
Example 4: Trading System, Trade Order Entry

On-line Trading System:
Trade Entry Screen

Core Features
- New Trade Order

Crosscuts

1. CN – Connectivity
2. DF-Out – Dataflow Out
3. FV – Field Validation
4. DDD – Data-Driven Defaults



Example 5: Trade Entry Screen (fragment)

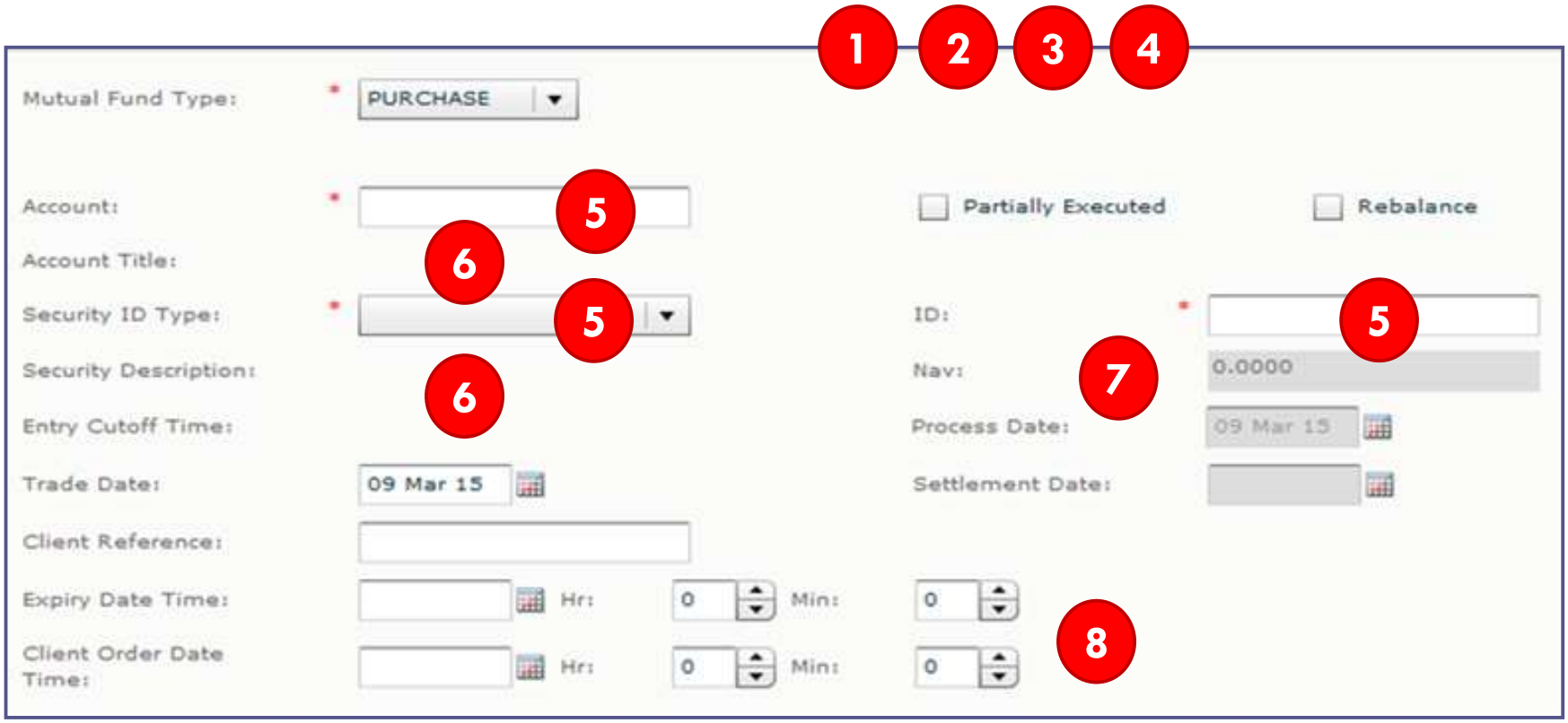
Trade Entry Screen

Core Features
- Create Trade
- Edit Trade

Crosscuts

1. ET – Entitlements
2. CN – Connectivity
3. CC – Concurrency
4. DF-Out – Dataflow Out
5. FV – Field Validation
6. DDD – Data-Driven Defaults

Example 6: Fund Trading Screen (fragment)



Core Features

- Create Trade
- Edit Trade
- Cancel Trade

Crosscuts

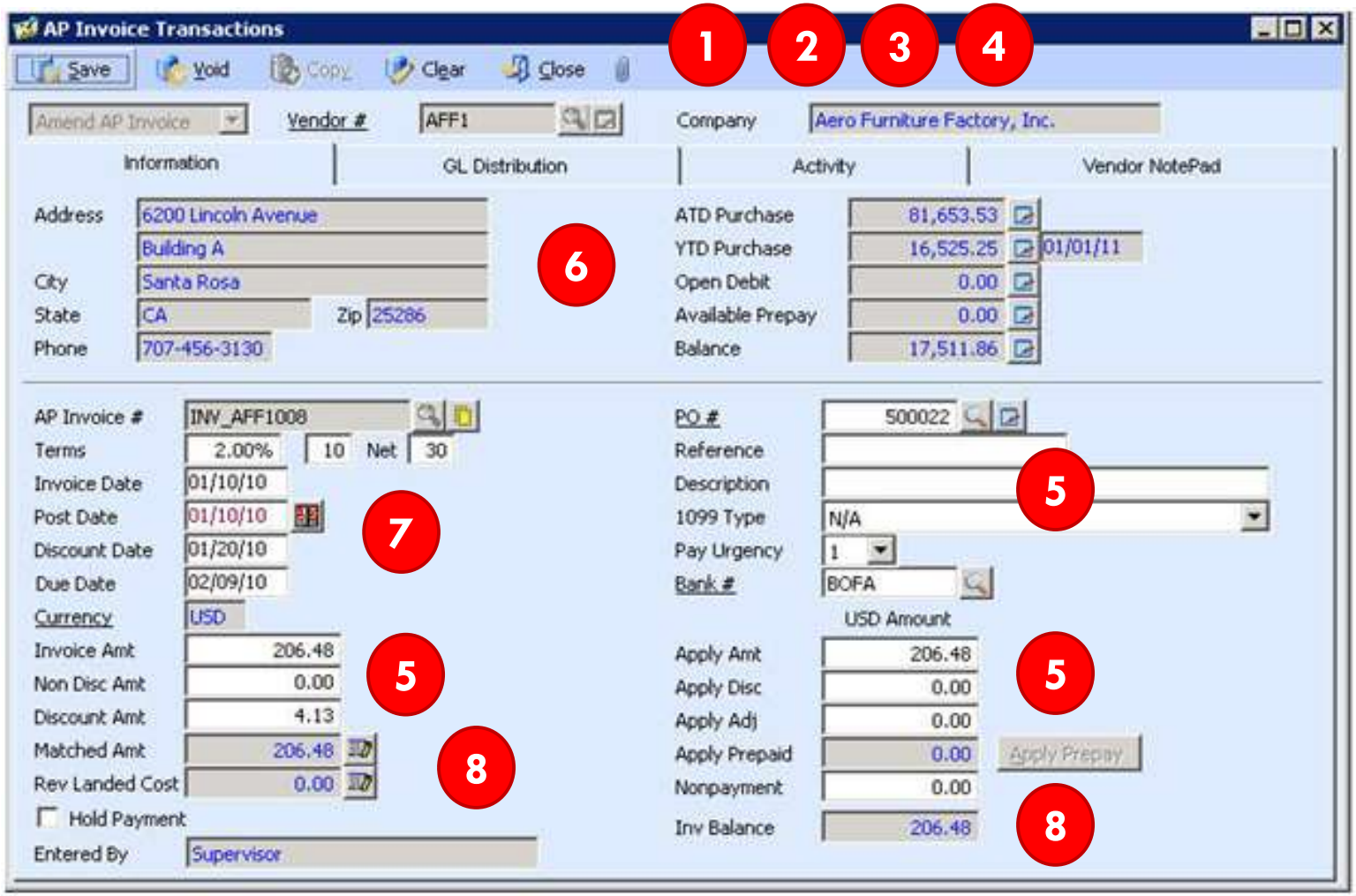
1.ET	3.CC	5.FV	7.SI-In
2.CN	4.DF-Out	6.DDD	8.DDV

Example 7: Accounting Software System

Accounts Payable Module

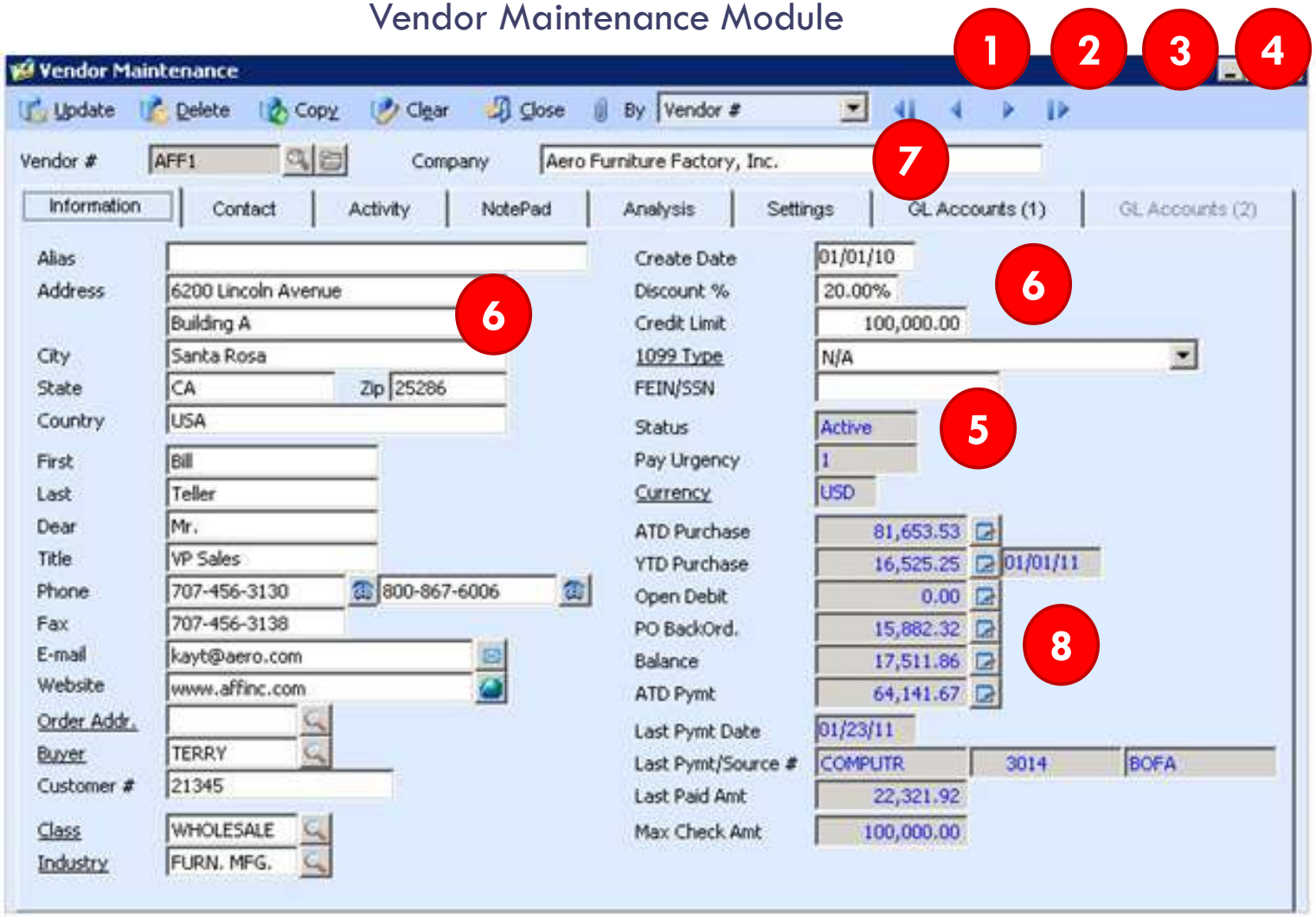
Core Features
- New Invoice
- Amend Invoice
- Cancel Invoice

Crosscuts
1. ET
2. CN
3. CC
4. DF-Out
5. FV
6. DDD
7. DDV
8. CL



Example 7: Accounting Software System

Vendor Maintenance Module



Core Features

- New Vendor
- Update Vendor
- Delete Vendor

Crosscuts

1. ET
2. CN
3. CC
4. DF-Out
5. VST – Vend. Status
6. FV
7. DDD
8. CL

Example 7: Accounting Software System

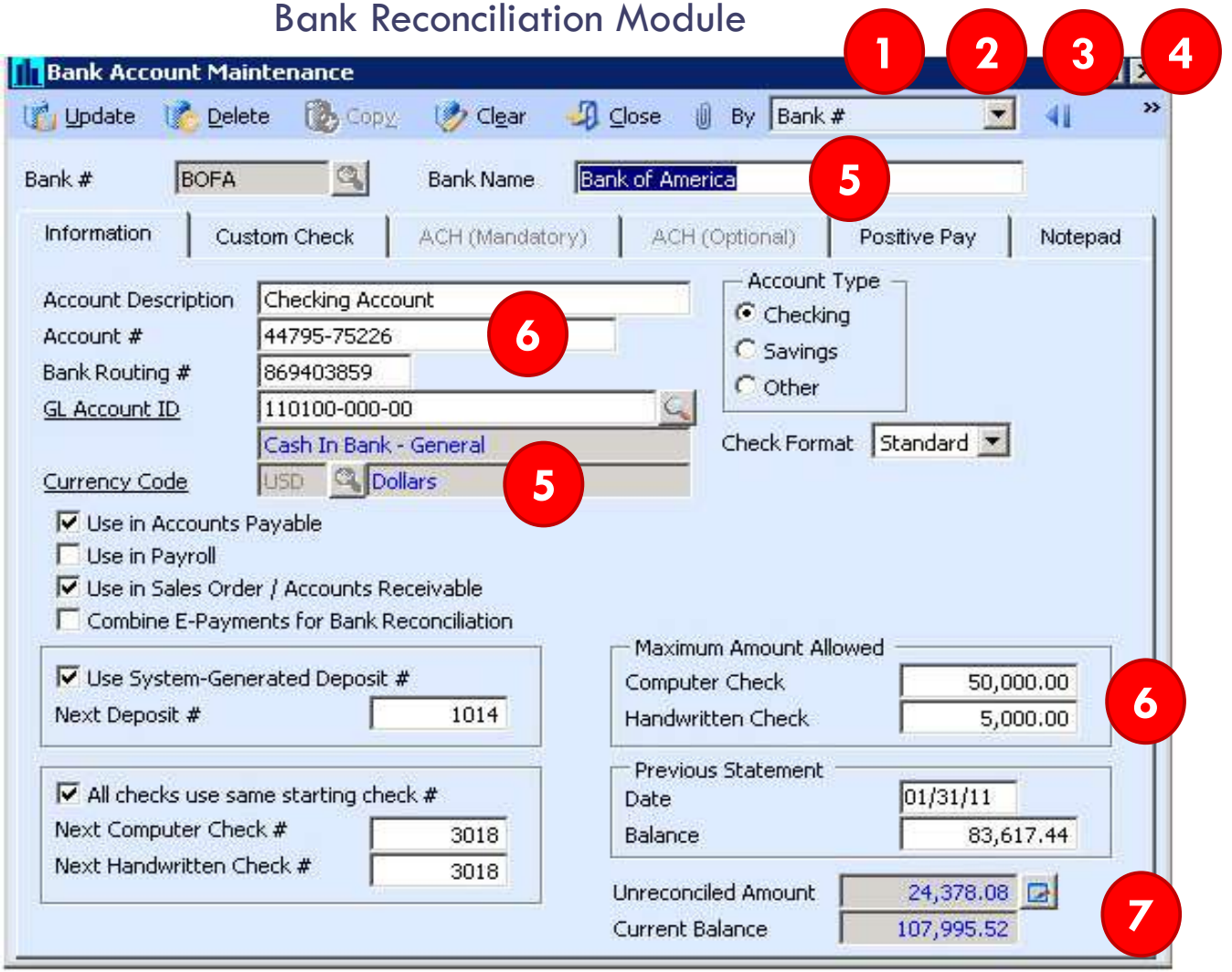
Core Features

- New Bank Account
- Update Bank Account
- Cancel Bank Account

Crosscuts

1. ET – Entitlements
2. CN – Connectivity
3. CC – Concurrency
4. DF-Out – Dataflow Out
5. DDD – Data-Driven Defaults
6. FV – Field Validation
7. CL – Calculation

Bank Reconciliation Module



Appendix B.

RCT Frequently Asked Questions

RCT's Frequently Asked Questions

1. Is the RCT technique intended to be used for existing production systems or new application development?
2. Creating an RCT is conceptually the same as performing functionality reverse-engineering, which is typically a time-consuming task for any business application. What is different about the RCT technique that allows us to perform the same task with a relatively small effort?
3. What is the difference between an **RCT** and **RTM** (Requirements Traceability Matrix)?
4. Who are the more common SMEs involved in the development of an RCT, e.g. end-users, business analysts, developers, or testers?
5. How do you know whether a newly-developed RCT is complete?
6. Can we automate the RCT development?

RCT's Frequently Asked Questions

7. Is the composition of concerns always "black & white" (i.e., 0 or 1)?
8. What can you do if you are not sure whether a crosscutting concern impacts a particular core feature?
9. Can a core feature not be impacted by any of the crosscutting concerns?
10. What if a given crosscutting concern does not impact any core features?
11. If two crosscutting concerns have the same composition pattern (0/1) and impact the same core features, can we merge them?
12. What is a practical number of crosscutting concerns for a business application? What if the list of crosscuts grows too long?
12. What is a Join Point? How can we use it?

Appendix C.

Examples of Crosscutting Concerns Common to Investment Banking Applications

Examples of Crosscutting Concerns

Concern Category	Concern Description	Type of Impact
ET-In - Internal Entitlements	This concern relates to different internal user access privileges (roles) and how they impact the behavior of core features. Examples: - Front Office Trader - Read Only - REPO Group - Price Group - Trade Support	Imposes a constraint on core feature execution
ET-Ex - External Entitlements	This concern relates to different external user access privileges (roles) and how they impact the behavior of core features. Examples: - Tier1ClientApprover - Tier2ClientApprover - ThirdPartyLevel1Approver - ThirdPartyLevel2Approver	Imposes a constraint on core feature execution
CS - Client Setup	This concern is used to model the dependency of core feature behavior on the client profile, client rules, etc.	Adds detail to a core feature context

Examples of Crosscutting Concerns

Concern Category	Concern Description	Type of Impact
PT - Product Type	This concern captures various financial product types that are used by core features of investment banking applications. Depending on the selected product type, functionality of the same core feature might be different. Examples of Product Types: - Equity - Fixed Income - MBS - Options - Repos - FX	Adds detail to a core feature context
TST - Trade Status	This concern relates to the trade statuses comprising the trade lifecycle. Examples: - VALIDATED - SENT - RECEIVED - RESEND, etc.	Imposes a constraint on core feature execution
FV - Field Validation	This concern relates to validating data entry for individual fields.	Interrupts the core feature flow

Examples of Crosscutting Concerns

Concern Category	Concern Description	Type of Impact
DDV - Data-Dependency Validation	This concern relates to validating field value combinations. It also includes validation of constraints, for example, a currency cannot be made inactive if it has outstanding orders/trades.	Interrupts the core feature flow
DDD - Data-Driven Defaults	This concern relates to populating field default values based on another field value. Examples: - Security description is populated based on the selected security symbol. - A dropdown list of trading books is populated based on the selected trader name, etc.	Adds detail to a core feature context
CL - Calculations	This concern represents various "behind-the-screen" calculations that are executed in the context of core features.	Adds detail to a core feature context
ER - Enrichment	This concern is commonly used in prime brokerage applications and relates to any data enrichment rules being applied to data coming from external clients, e.g., hedge funds.	Adds detail to a core feature context

Examples of Crosscutting Concerns

Concern Category	Concern Description	Type of Impact
CC - Concurrency	This concern relates to simultaneous [transaction] data manipulation by two or more users. For example, both users see the same trade in the blotter. User 1 tries to amend the trade, whereas User 2 tries to cancel the same trade at the same time.	Interrupts the core feature flow
CN - Connectivity	This concern relates to the broken architecture of an application where the front-end can be disconnected from the back-end and that might change the behavior of the impacted core features. Commonly, front-office trading systems implement this functionality.	Interrupts the core feature flow
RG - Region	This concern relates to Global applications, i.e., used in different regions. Some core features may behave differently depending on the region where they are used. Regions are: - Europe - Americas - APAC	Adds detail to a core feature context

Examples of Crosscutting Concerns

Concern Category	Concern Description	Type of Impact
DF-In - Data Flow In	This concern addresses the core feature behavior when it takes data in from another core feature of the same application.	Adds detail to a core feature context
DF-Out - Data Flow Out	This concern addresses the core feature behavior when it generates data used by other core features of the same application.	Adds detail to a core feature context
SI-In - System Interface (inbound)	This concern relates to receiving [transaction] data from external applications and is used to indicate which of the core features of a given application consume such data. Using this concern category can help a team to better analyze the impact of changes and better plan E2E testing.	Adds detail to a core feature context
SI-Out - System Interface (outbound)	This concern relates to sending [transaction] data from core features of a given application to external applications. Using this concern category can help a team to better analyze the impact of changes and better plan E2E testing.	Adds detail to a core feature context

Examples of Crosscutting Concerns

Concern Category	Concern Description	Type of Impact
TH - Trade History	Commonly, trading applications capture trade history and allow users to audit and see what actions were applied to a given trade to date.	Adds detail to a core feature context
CA - Cache	Some static data on the back end can change during a day and may not be stored in the memory cash and immediately available for the front-end users. To retrieve this data, the user should explicitly refresh cache; after this action core features could process the latest values of static data.	Adds detail to a core feature context
PF - Performance	This concern relates to performance requirements applied to some core features.	Adds detail to a core feature context