A Scalable "Just Enough" Approach to Requirements

Andreas Gschwind
Rational Services Executive GMU

Rational. software

RDM02

Based on Ideas from Kurt Bittner/Robin Bater

Go to IBM

Topic: Adopting an Agile approach to Requirements

- From engagements with our clients
  - We don’t do requirements any more
  - Backlog based
  - Test case driven
  - Feedback loop
  - Scalability ?
    - Complex systems
    - Distributed teams
    - Team size > 5

The Problem with Software

- Organization's goals are poorly understood by IT, & often by the business itself
- "Requirements" processes often dive too deep, too early into specifying poorly conceived solutions
- Cultural divide between organizations clouds communication
- Exploring innovative alternatives is slow & expensive

- How to overcome? Facilitate discussion

Some Interesting things about Requirements

Users expect functionality they did not initially ask for	93%
Requirements are incomplete	89%
Requirements are unclear or ambiguous	85%
Developers make assumption when they encounter ambiguities or gaps	85%
Users demand functionality that they never use	78%
Models are not consistently used or not used well	74%
The project's vision and scope are not clearly defined	74%
Internal customers are unhappy due to project delays and missing the mark	67%
Models are not consistent with written requirements	59%
Requirements are contradictory or conflicting	52%

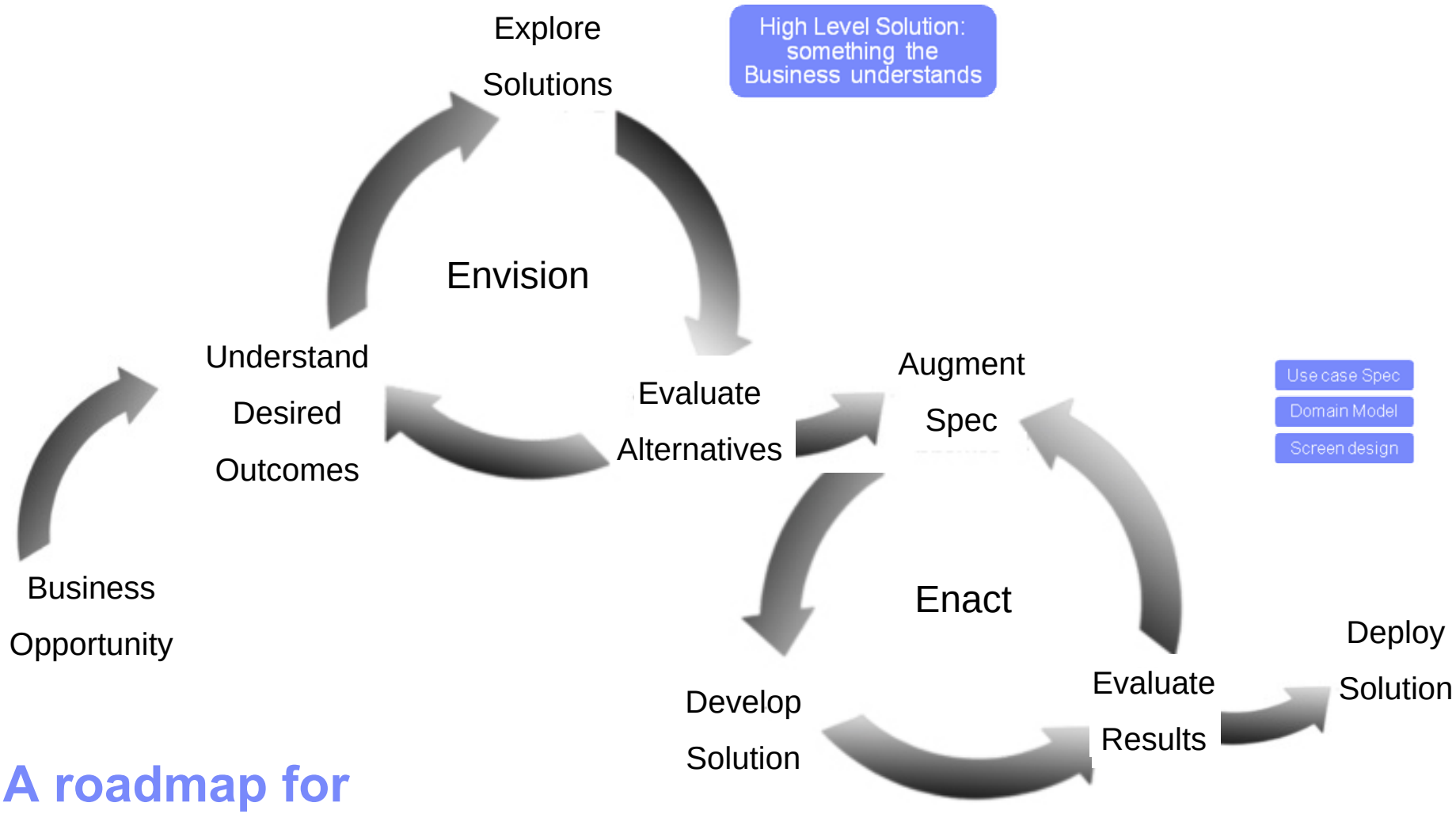
Source: Dr. Dobb's Requirements Development Journal, 2006

Have a discussion to find out what the business needs

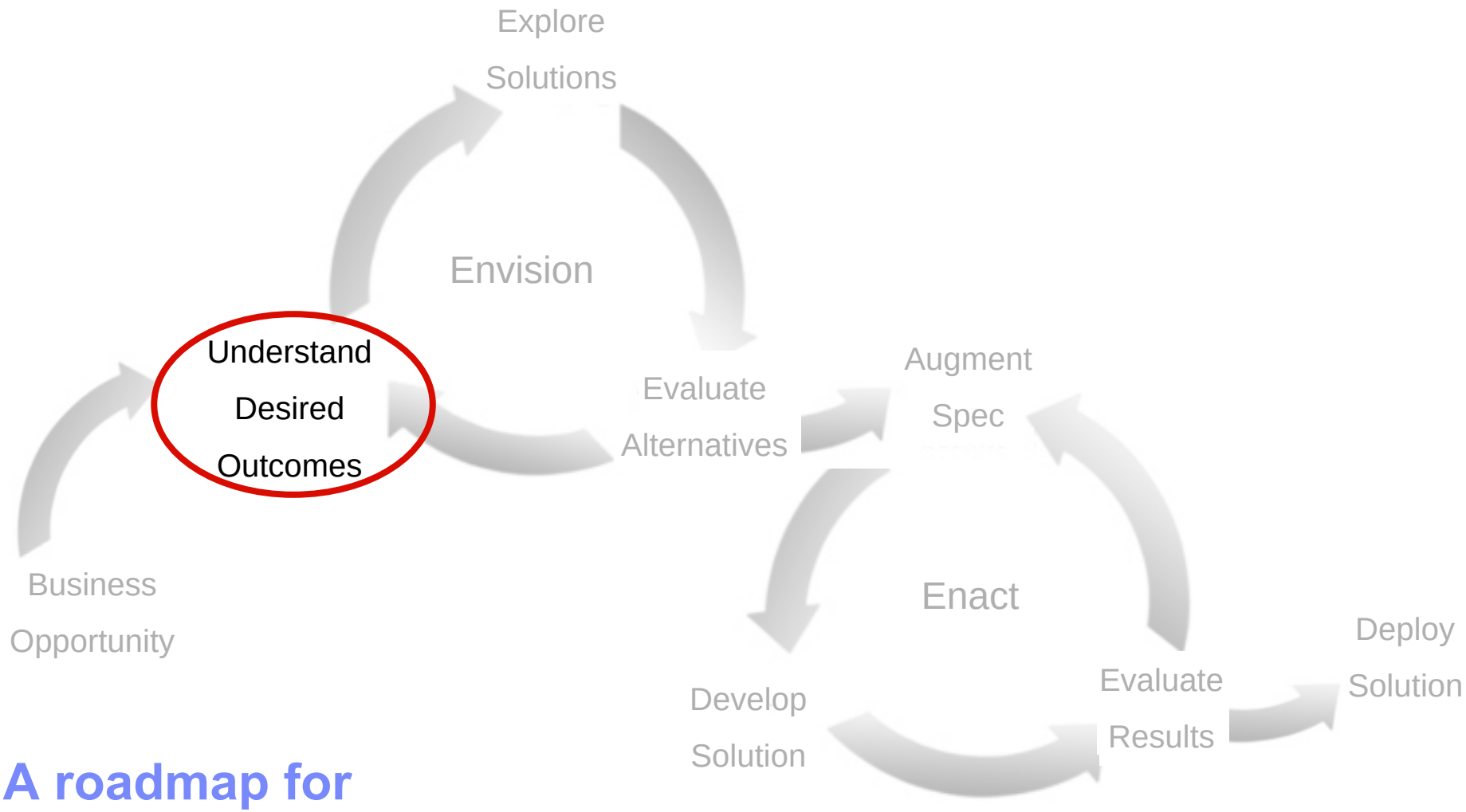
- Next generation airplane

- Boing: smaller, more hops, less travellers per hop, more routes
- Airbus: bigger, longer distance, direct connection

- Business doesn't always understand what they want



A roadmap for building the right solution



A roadmap for building the right solution

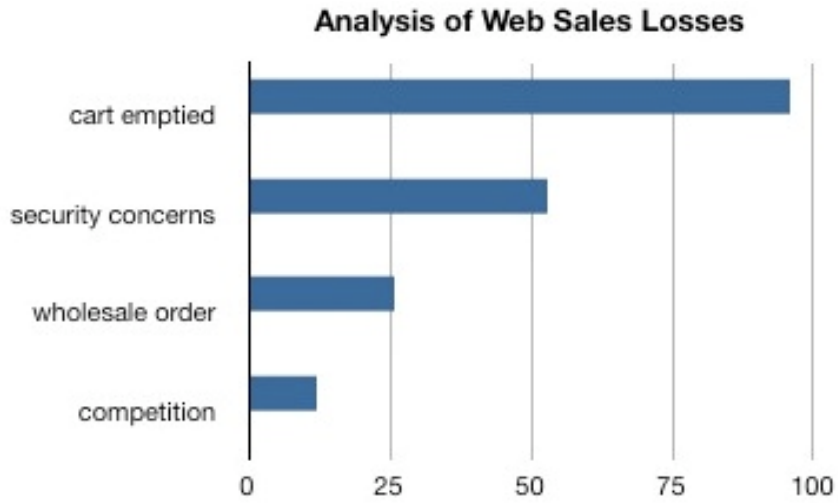
Understanding Desired Outcomes

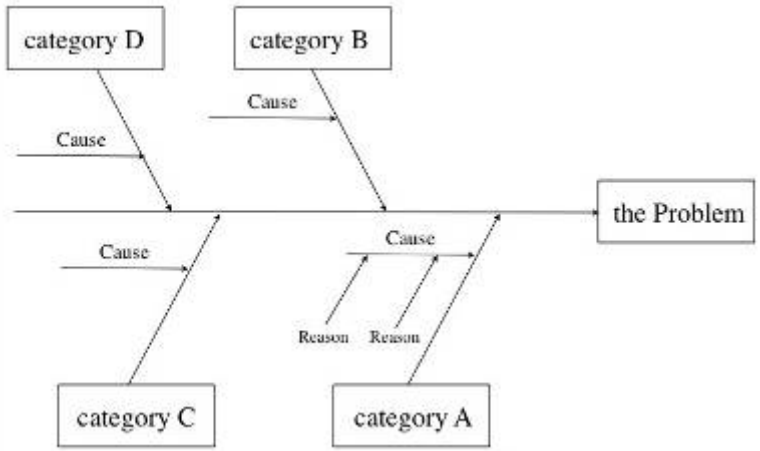
- The business opportunity is rarely well understood, even by the business
  - The business benefits are usually overstated and expected costs are understated
  - Real needs are obscured behind “wish lists”
- What you *really* need to understand are desired outcomes
  - Ask what they want to achieve, how they will know if they have achieved it
  - The solution they have asked for may or may not deliver the desired outcomes
  - Business result

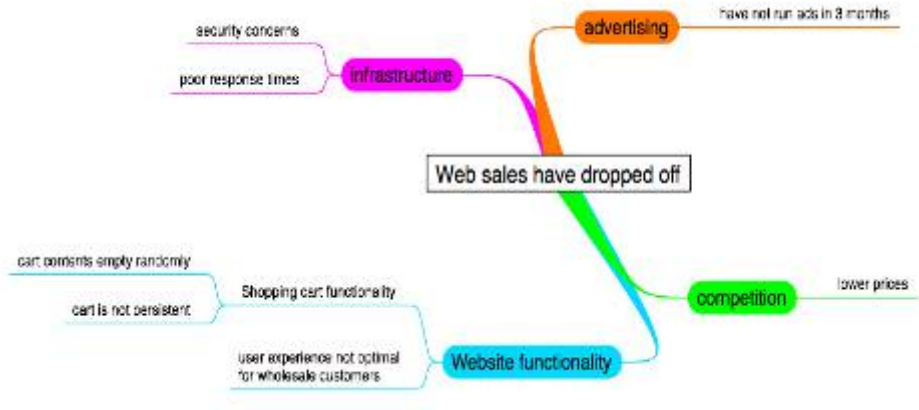
You need to get to the root of the problem, not just treat the symptoms

Some Techniques for Understanding Needs

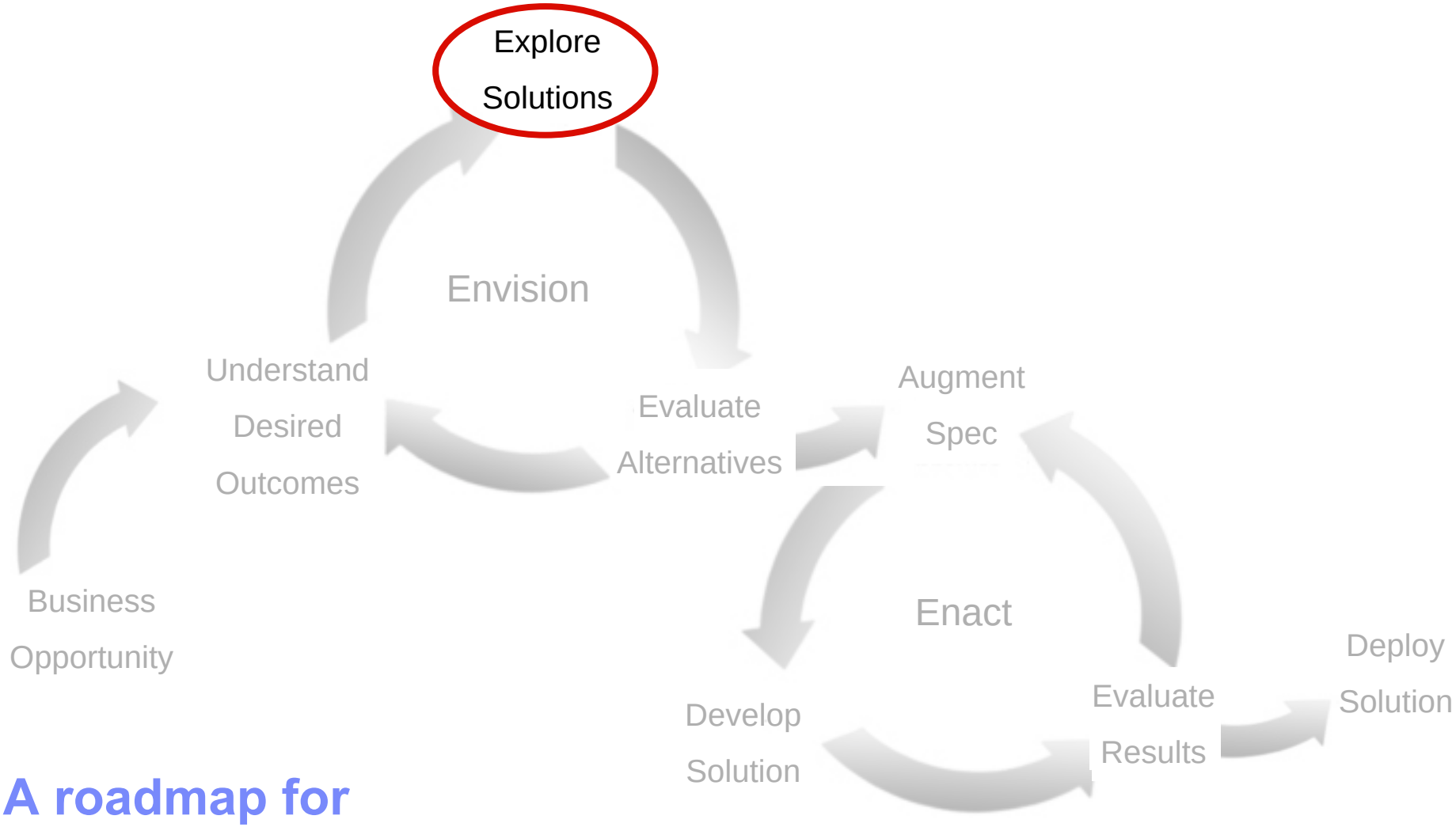
- 5 Whys
- Fishbone diagrams
- Pareto Charts
- Mind Maps
- Desired Outcomes Analysis







What are you trying to achieve vs what do you need



A roadmap for building the right solution

Requirements as a Means of Communication

Use-Case Specification – Register for Courses

Brief Description

This use case allows a Student to register for course offerings in the current semester. The Student can also modify or delete course selections if changes are made within the add/drop period at the beginning of the semester. The Course Catalog System provides a list of all the course offerings for the current semester.

Actors

1. *Primary Actor – Student*
2. *Secondary Actor - Course Catalog System*

Flow of Events

1. *Basic Flow*

1.1. LOG ON.
This use case starts when a student accesses the Course Registration System. The student enters a student ID and password and the system validates the student.

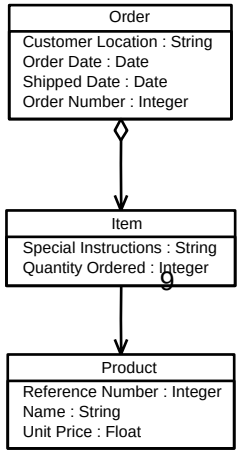
1.2. CREATE SCHEDULE.
The system displays the functions available to the student. These functions are: Create A Schedule, Modify a Schedule and Delete a Schedule. The student selects 'Create a Schedule'.

1.3. SELECT COURSES
The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student. The Student selects up to 4 primary course offerings and 2 alternate course offerings from the list of available offerings. The student can add and delete courses as desired until choosing to submit the schedule.

1.4. SUBMIT SCHEDULE.
The student indicates that the schedule is complete. The system validates the courses selected and displays the schedule to the student. The system displays the confirmation number for the schedule. The systems saves the student's schedule information. The use case ends.

Use-Case Specification

but also:



Domain Model

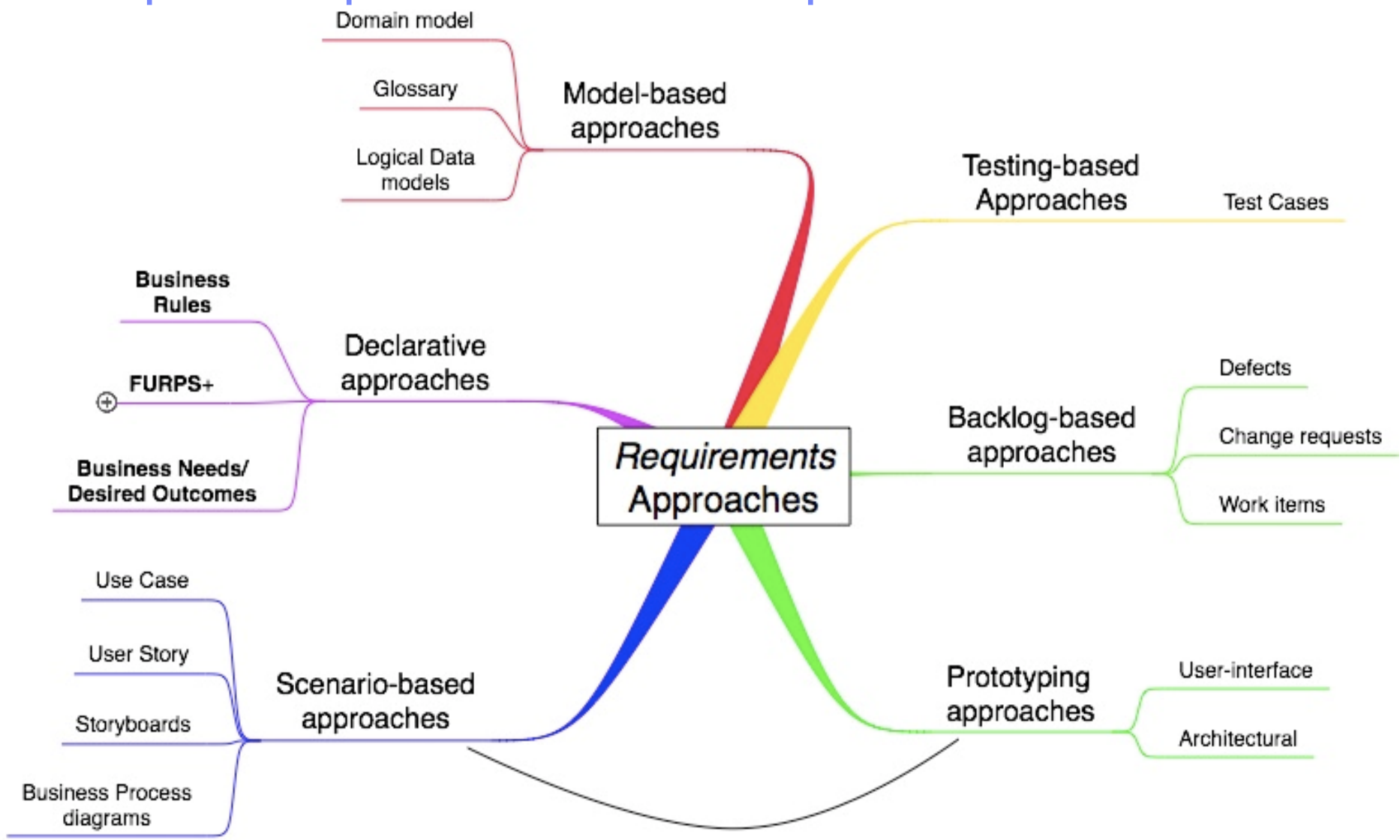
and:

User Interface Prototypes and/or Storyboards

as well as:

- Glossary of Terms
- Supplementary (declarative) Requirements
- Business Rules
- Test Cases
- ...

A Map of Requirements Techniques



Comparing Requirements Techniques

Technique	Facilitates discussion	Handles inter-dependencies	Low over-head	Handles "flows"
Declarative (Traditional)	Moderate	Poor	Moderate	Poor
Scenario-based	Good	Moderate	Poor	Good
Model-based	Moderate	Good	Poor	Moderate
Prototype-based	Good	Poor	Moderate	Good
Testing-based	Moderate	Poor	Good	Moderate
Backlog-based	Poor	Poor	Good	Poor

- No technique works best in all cases
- Too much detail is as bad as too little!
- The best approach is a blend of techniques

An Agile approach to requirements

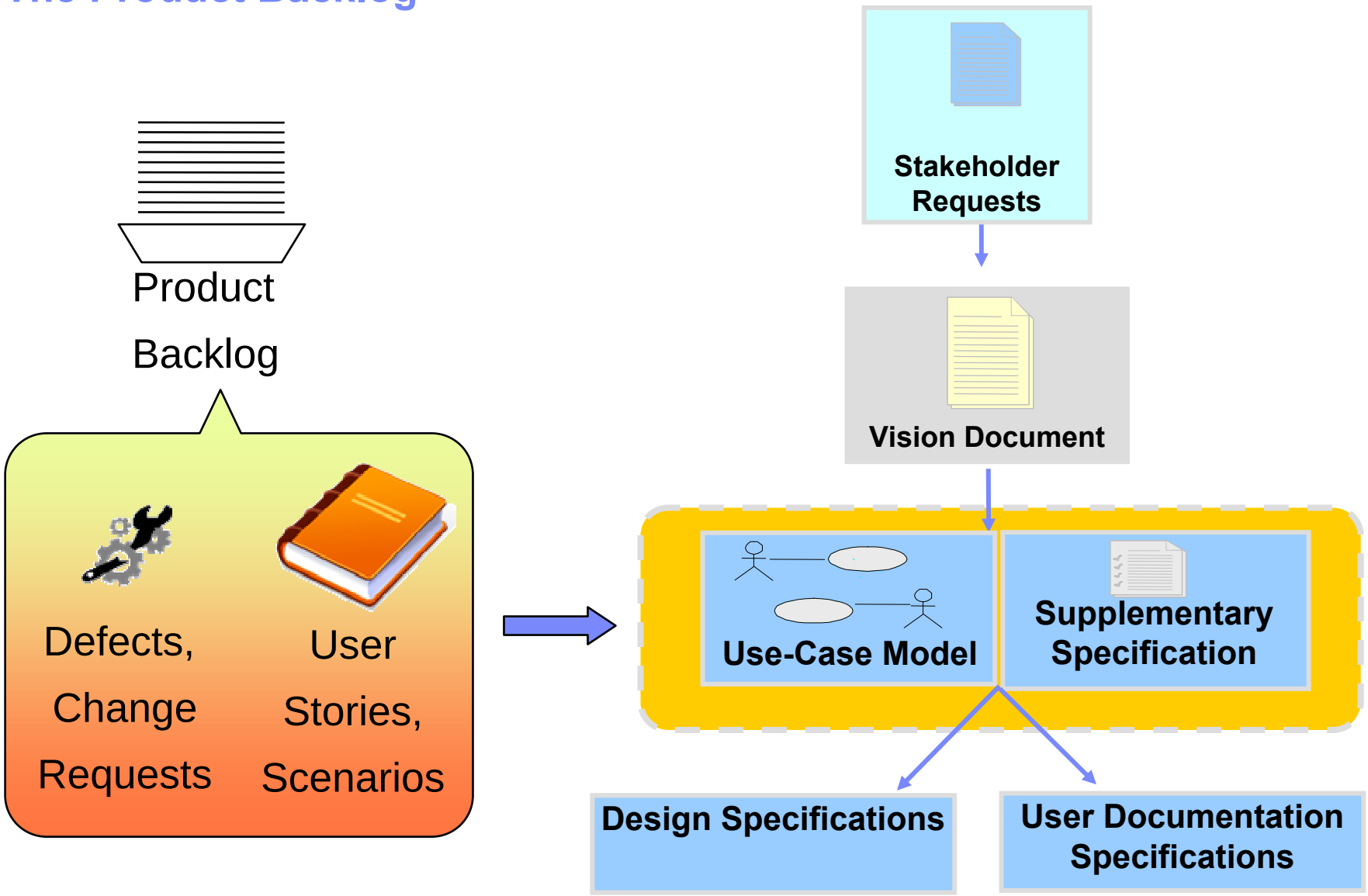
1. **Start with a Product Backlog**
2. **For *major* new functionality, identify use cases**/stories and scenarios
3. **Outline the flows** (if there are flows)
4. **Define unfamiliar terms in a glossary**

- **Augment the Glossary with a Domain Model** to describe concepts with relationships or structure

5. **Prototype** the visualizable
6. **Add declarative requirements** for things that can't be expressed in scenarios, prototypes or as backlog items
7. **Write detailed descriptions of scenarios** only where the need for precision justifies the investment in effort

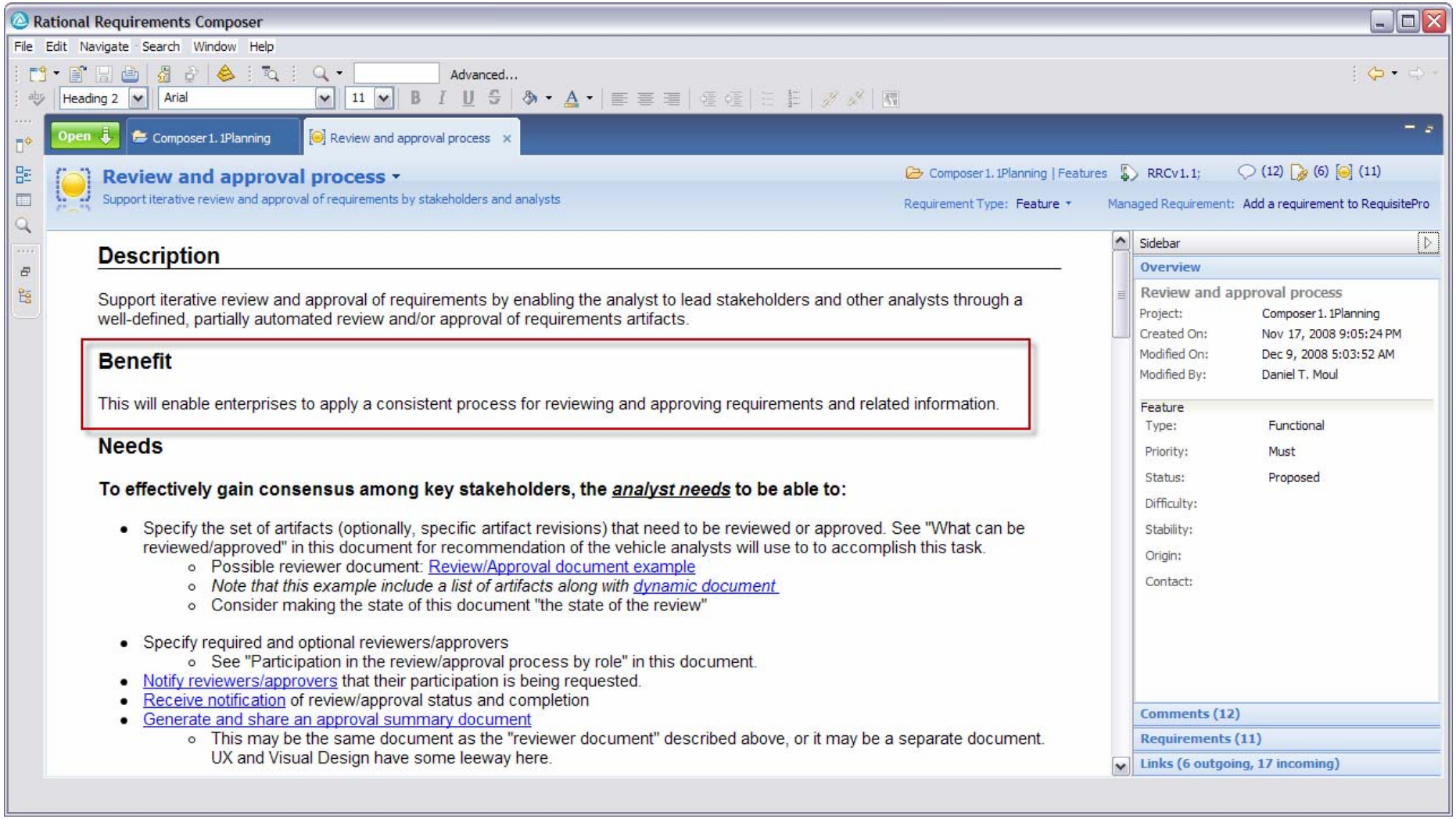
Incremental & iterative, stop when you think you have enough info

1) The Product Backlog




Home
About Jazz
Projects
Downloads
Forums
Library
Community
Rational Team Concert
Robin Bater
Log Out
Dashboards
Project Areas
Work Items
Plans
Builds
Reports
Rational Requirements Compo
Release Requirement 13783
Save
Summary:
Review and Approval
Triaged
Create
Work Item
Create Query
Overview
Links
Approvals
History
Loaded: Jul 21, 2009 4:01 P.M.
Details
Type:
Release Requirement
Time Spent:
Quick Information
Subscribers (1): SVD
Children (13): 13408, 13508, 13844, 13848, 13849, 13857, 13861, 13862, 13863, 13866, ...
Depends On (1): 14912
Duplicated By (1): 8347
Related (3): 15676, 15677, 15678
Related Artifacts (2)
Recently Viewed
Release Requirement 13783
My Queries
Drink Our Champagne
RequisitePro Integrations
Shared Queries
Predefined
Closed created by me
Closed subscribed by me
Milestone Scenario
New unassigned
Open assigned to me
Open assigned to me (current milestone)
Open created by me
Composer [Composer Main]
ComposerFixpack1 [Composer 1.0 Fixpack 1]
ComposeriFix1
Children (13)
Enhancement 13408: Feature: Review and Approval
Enhancement 13508: Feature: Review and Approval
Milestone Requirement 13844: Review and Approval/Define resource format for a review
Milestone Requirement 13848: Review and Approval/Create a review
Milestone Requirement 13849: Review and Approval/Edit a review
Milestone Requirement 13857: Review and Approval/Authorization considerations for reviews and approvals
Milestone Requirement 13861: Review and Approval/Participate in a review
Milestone Requirement 13862: Review and Approval/View a review (participant view)
Milestone Requirement 13863: Review and Approval/Notifications for reviews and approvals
Milestone Requirement 13866: Review and Approval/Link to a review
Milestone Requirement 13877: Review and Approval/Implement state transitions for a review
Enhancement 15244: Initial support for reviewing Collections
Milestone Requirement 16043: Migrate Review APIs to use new Server APIs
Edit
Save

New Functionality Description

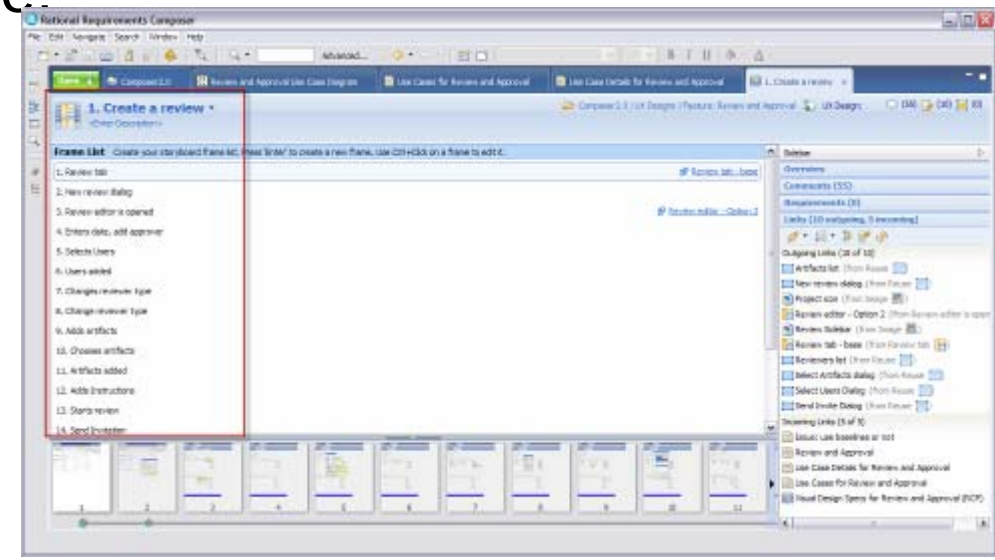


2) Use Cases and User Stories

- User Stories are statements made by the customers as things that the system needs to do for them
  - They are in the format of about three sentences of text written by the customer in the customer's terminology without techno-syntax.
  - User stories provide a good way to start talking about what the system needs to do
- User stories often correspond to *identified* "scenarios" of a use case
  - Given a number of user stories, it is often possible to start synthesizing the use cases
  - Find the common parts of a number of user stories - this becomes the start of the "basic flow" of the use case
  - The leftover parts start to form the alternative flows
  - Refine both over time as new stories are found
- Large numbers of user stories become unmanageable; use cases provide a way to consolidate the user stories
  - With large numbers of stories it is hard to see what's in common

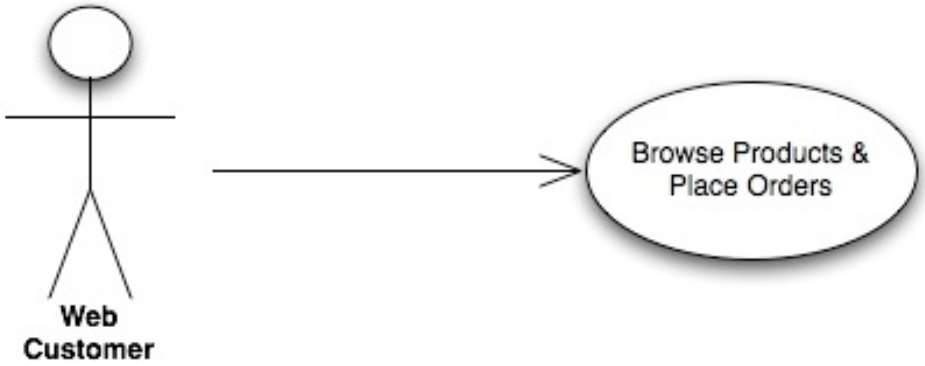
Example User Stories

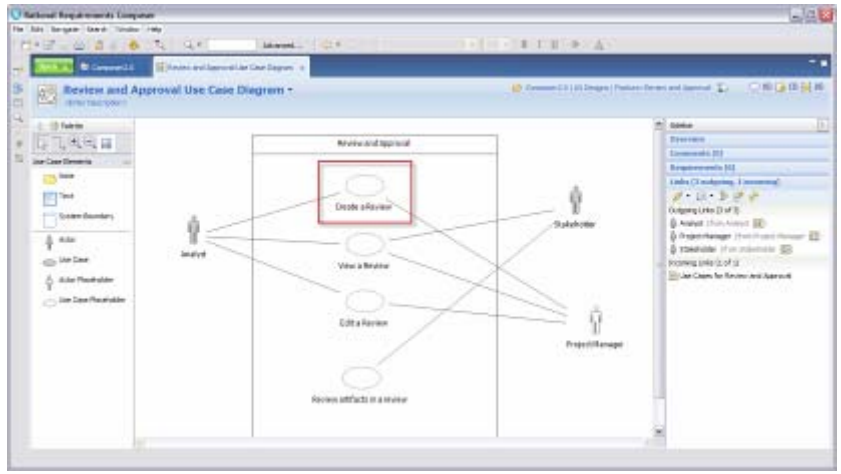
- Web Customers purchase products online.
- When attempting a purchase, the credit card may be rejected.
- Inventory is received.
- Inventory is reconciled.
- An online order is abandoned.
- An online order is interrupted.
- The shopping cart for an online order is saved for later.
- ...



3) Identifying Use Cases to Map to Desired Outcomes

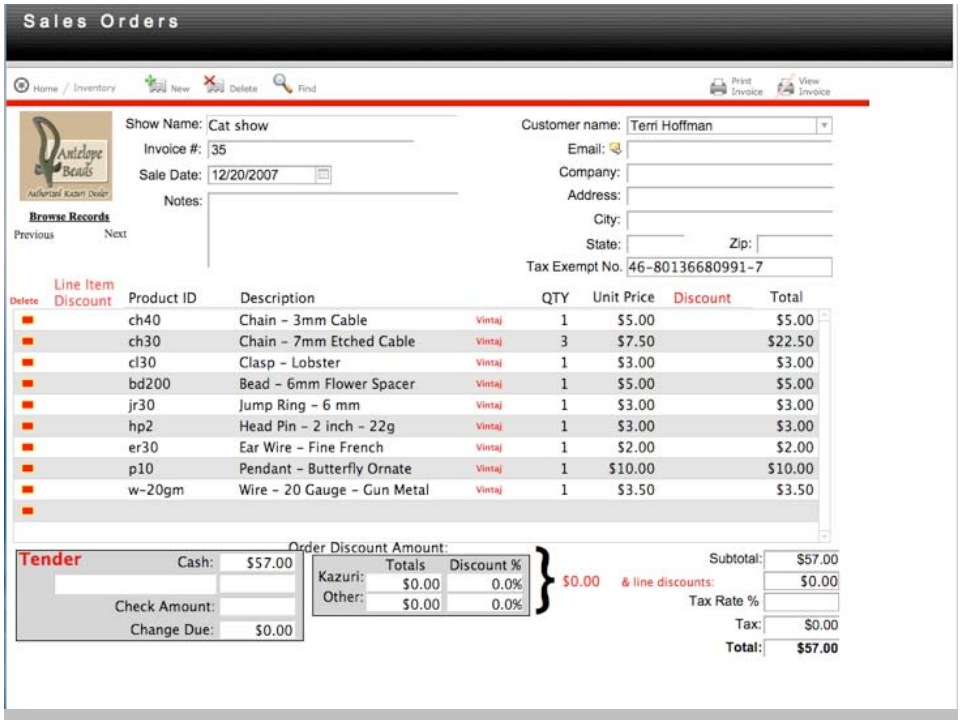
Example





This use case describes how a Web Customer uses the system to view and purchase the products on sale. Products can be found by various methods including browsing by product type, browsing by manufacturer or key word searches.

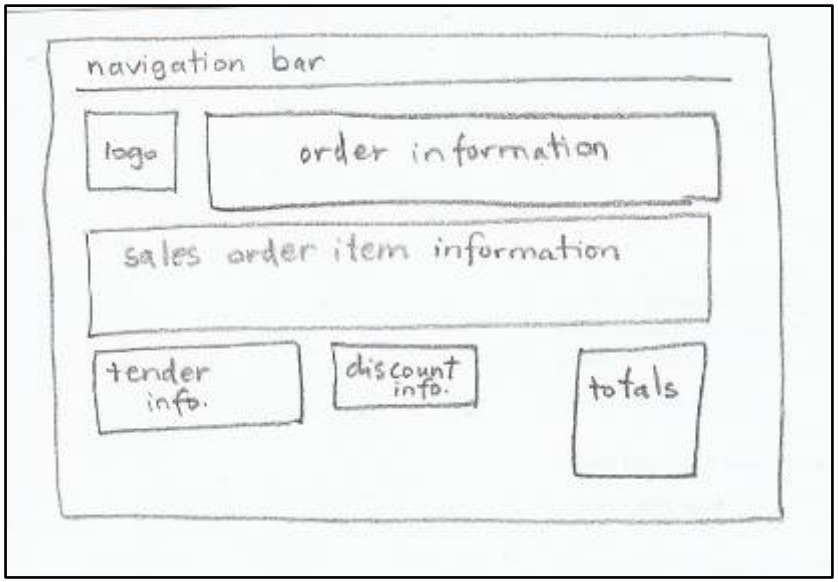
5) Exploring Solutions through Visualization



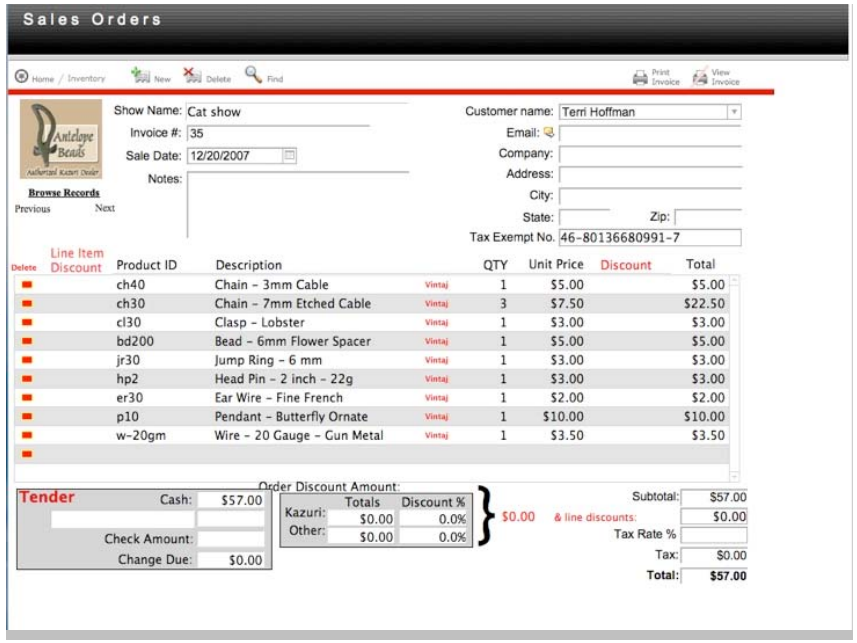
Visualization converges the vision of the solution faster

- Most people think visually rather than verbally; visualization enables people to be more creative
- Visualization forces people to be concrete about the solution that they often have in their heads
- Visualization creates more interesting discussions about possibilities than “documents”
- Visualization provides a way to articulate the solution in in a language everyone can understand

Examples of Visualization

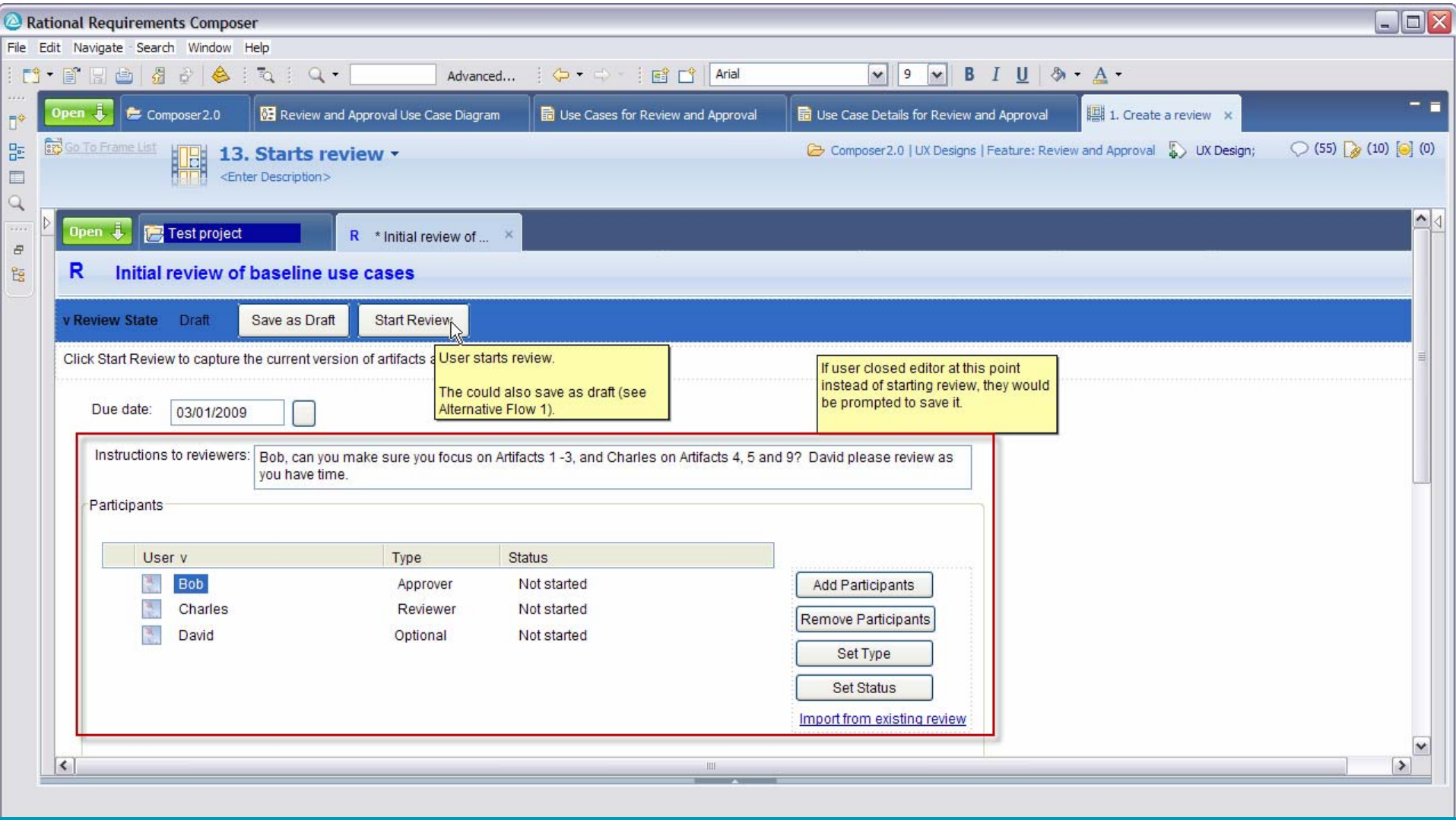


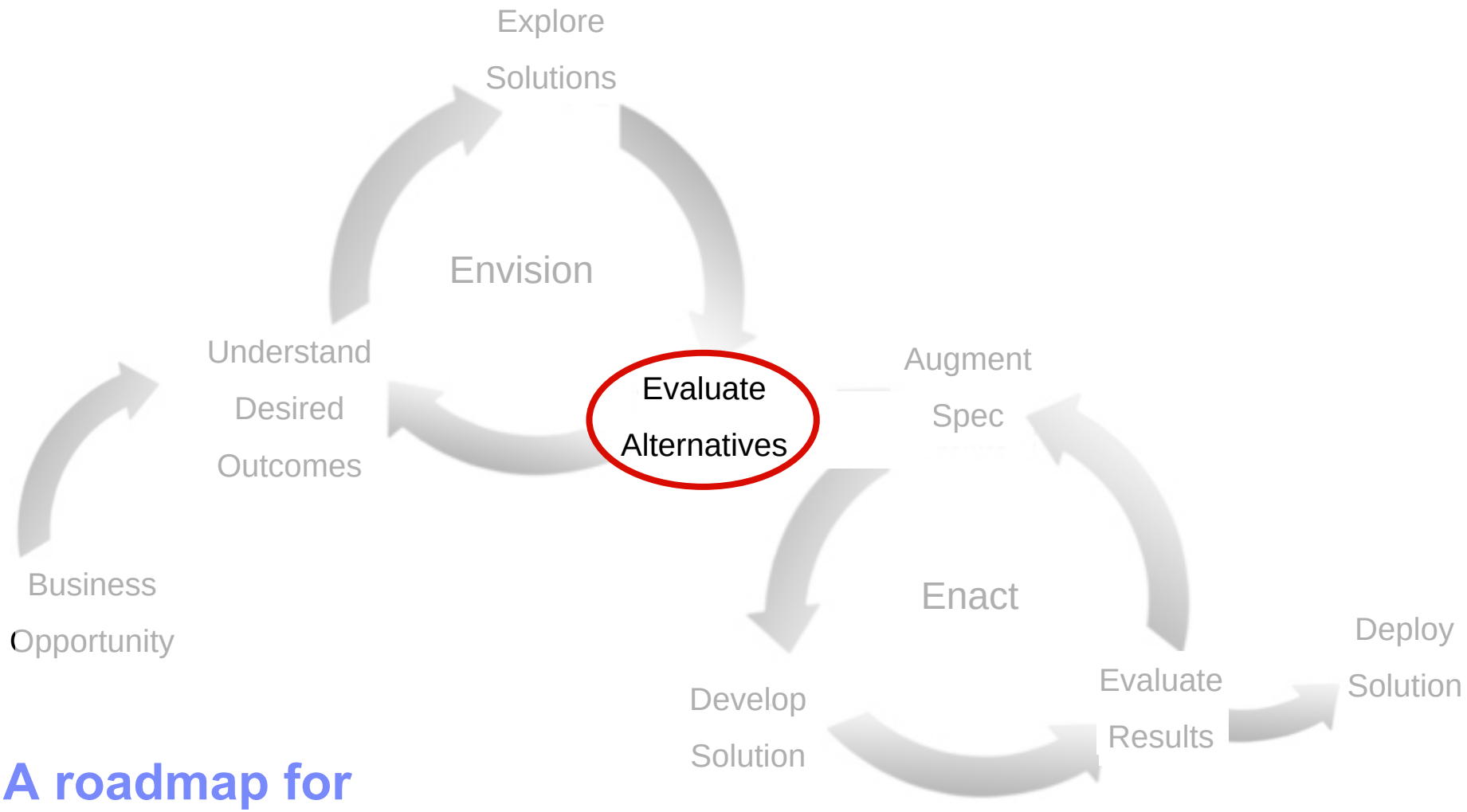
Informal Sketch



High-fidelity Prototype

Example UI Design in RRC





A roadmap for building the right solution

Strategies for Reviews

- Using the visualizations, walk through scenarios, describing how the solution will deliver the desired outcomes
- Bring in other reference material where needed to supply detail
- Make the sessions interactive
- Look for undiscovered desired outcomes or ways to improve processes
- Seek to simplify

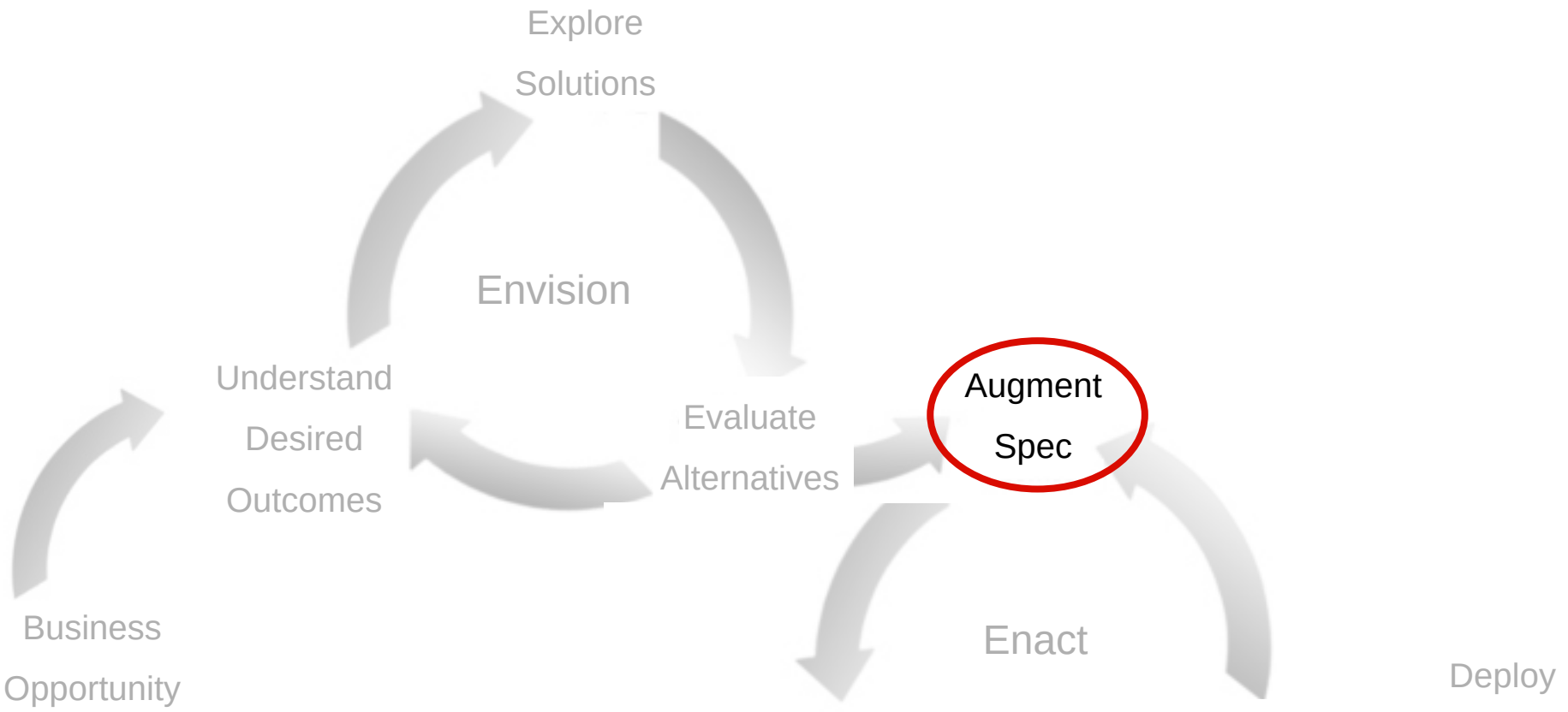
Don’t circulate documents as discussion basis!

Use Documents as final record on agreement

You won’t get good feedback and you’ll miss the opportunity for an interactive discussion about whether you’ve met the real needs.

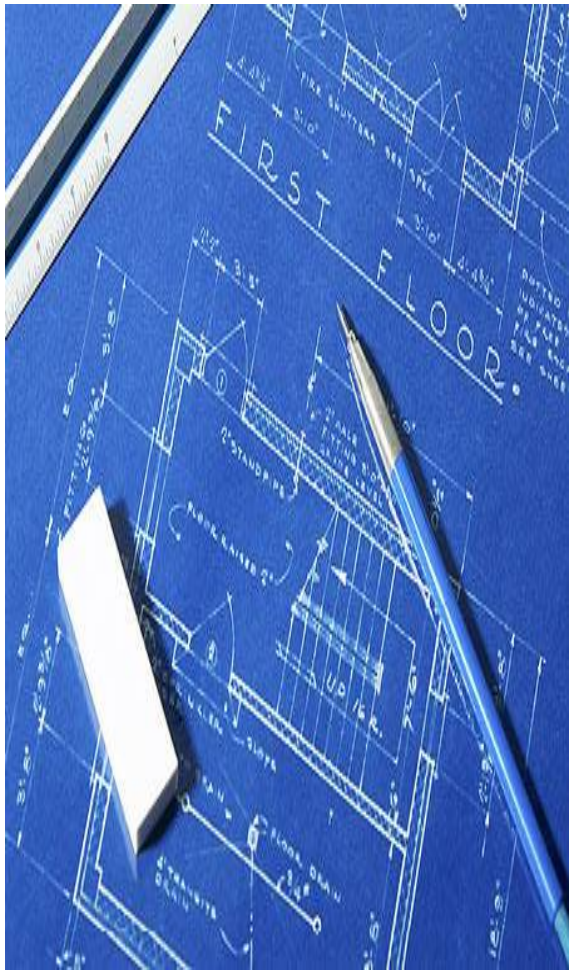
Choosing the Right Participants

- Select people with a deep understanding of the process being improved
  - People who will be directly affected by the outcomes produced by the solution
  - A small but diverse group of individuals – not just “lead users” but also, and especially, “average” users
- Avoid people only indirectly involved in the solution
- Avoid using review sessions for “information sharing” (Korean way)

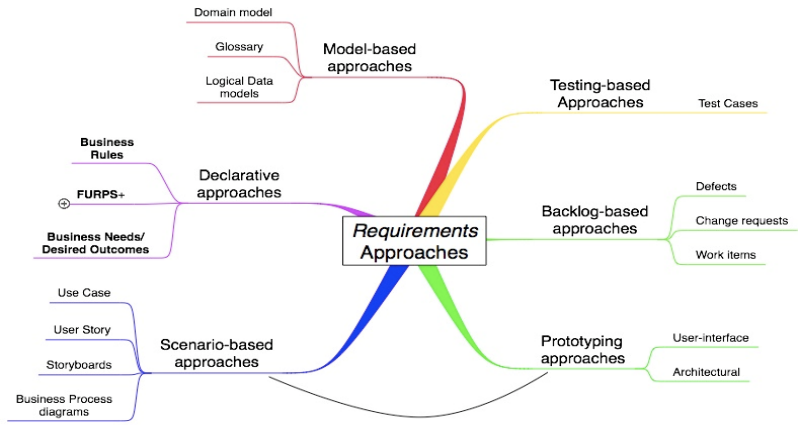


A roadmap for building the right solution

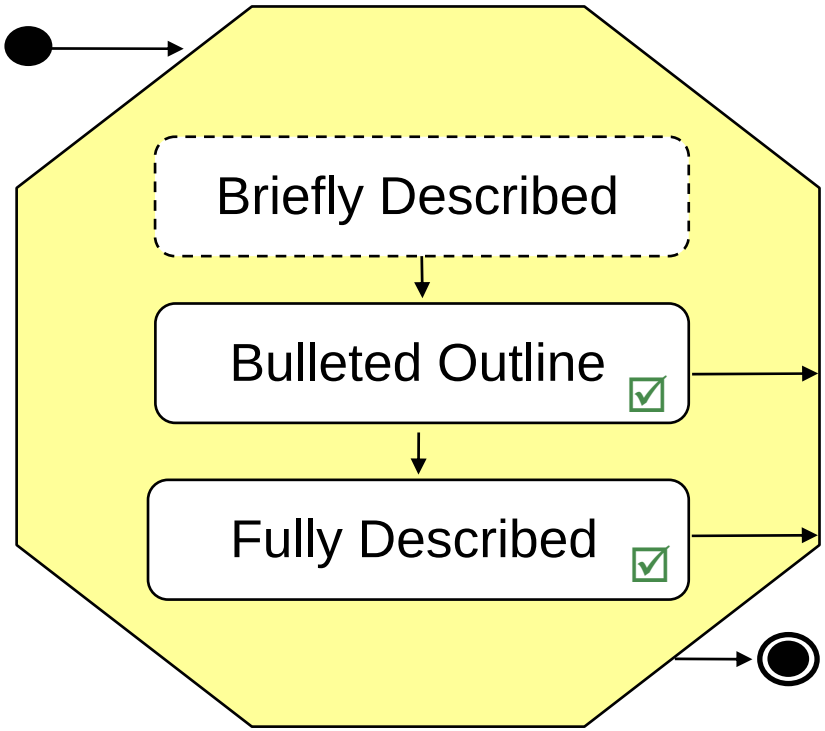
Augmenting Specifications



- *Envisioning* work will have created a rough picture of the solution; the work here is to add details, where needed, to create a specification of what needs to be built
- Some or all requirements techniques may be used to augment the items in the Product Backlog



Levels of Detail in Use Case Specifications: An Example



- Each *use case flow* may be at a different level of detail
- Many or most use cases may only be outlined if the behavior is simple
- Only the most complex flows need to be fully described

Exploiting the levels of detail

Authoring State	Primary Purpose	Supports
Briefly Described	Identify the use case and summarize its purpose	• Basic Scope Management • Discussions about requirements
Bulleted Outline	Summarize the shape and extent of the use case	• Scope Management • Low fidelity estimation • Collaborative test definition • Prototyping
Fully Described	Provide a full requirements specification for the behaviour encapsulated by the use case.	• High fidelity estimation •Analysis and Design • Implementation and testing • Creation of user documentation

Strategies for Deciding How Far to Take Artifacts

↑ Complex behavior ➔ more precision

↑ Importance of implementing in a specific way ➔ more precision

↑ Regulatory scrutiny ➔ more precision

↓ Faster time to market ➔ less precision

↓ Simpler behavior ➔ less precision

↓ Lower risk of misunderstanding what is needed ➔ less precision

Don't detail CRUD!, use prototypes & a domain model

Create, Retrieve, Update, Delete)

Bulleted Outline

Basic Flow

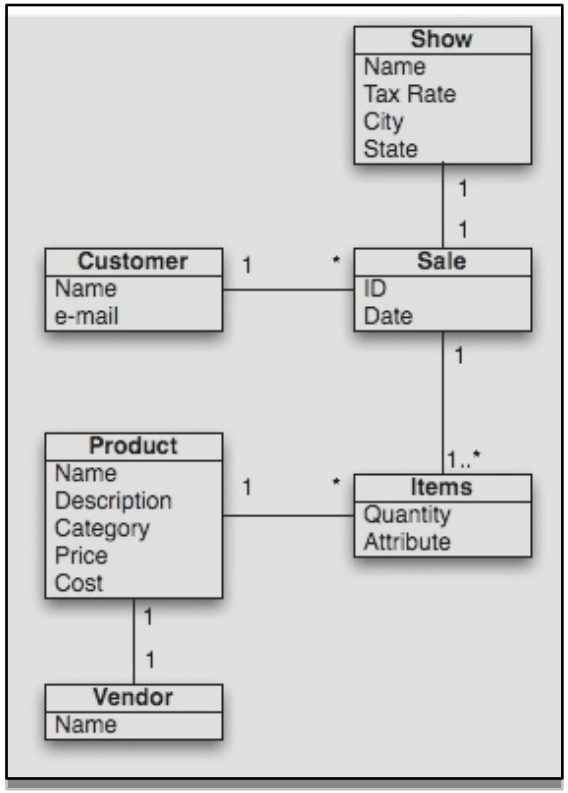
1. Browse Products
2. Select Products
3. Identify Payment Method
4. Identify Shipping Method
5. Confirm Purchase

Alternative Flows

A1	Key Word Search
A2	No Product Selected
A3	Product Out of Stock
A4	Payment Method Rejected
A5	Shipping Method Rejected
A6	Product Explicitly Identified
A7	Order Deferred
A8	Ship to Alternative Address
A9	Purchase Not Confirmed
A10	Confirmation Fails

Etc…

Using a Domain Model to capture Data Requirements



Items
Quantity
Attribute

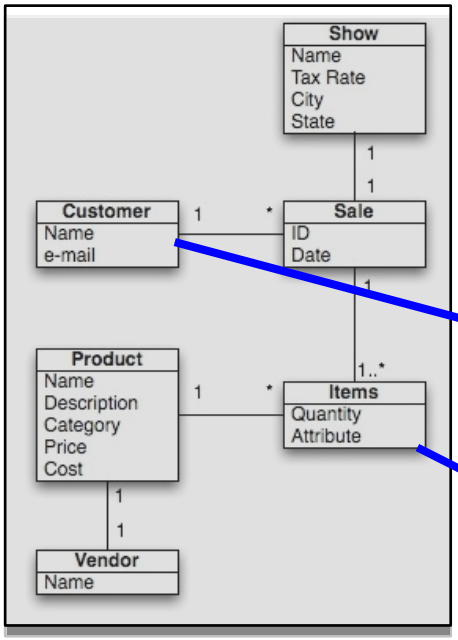
Data Rules:
Quantity > 0
Quantity ≤ Product.Quantity

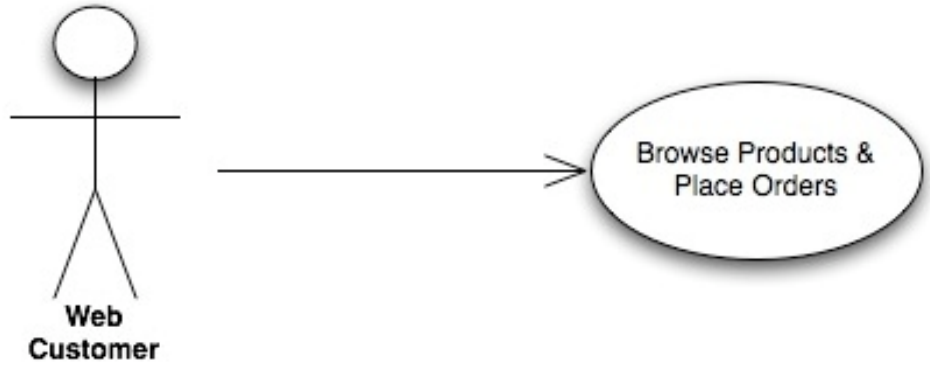
The Domain Model is a convenient mechanism for capturing requirements about data - think of it as an extension of the Glossary

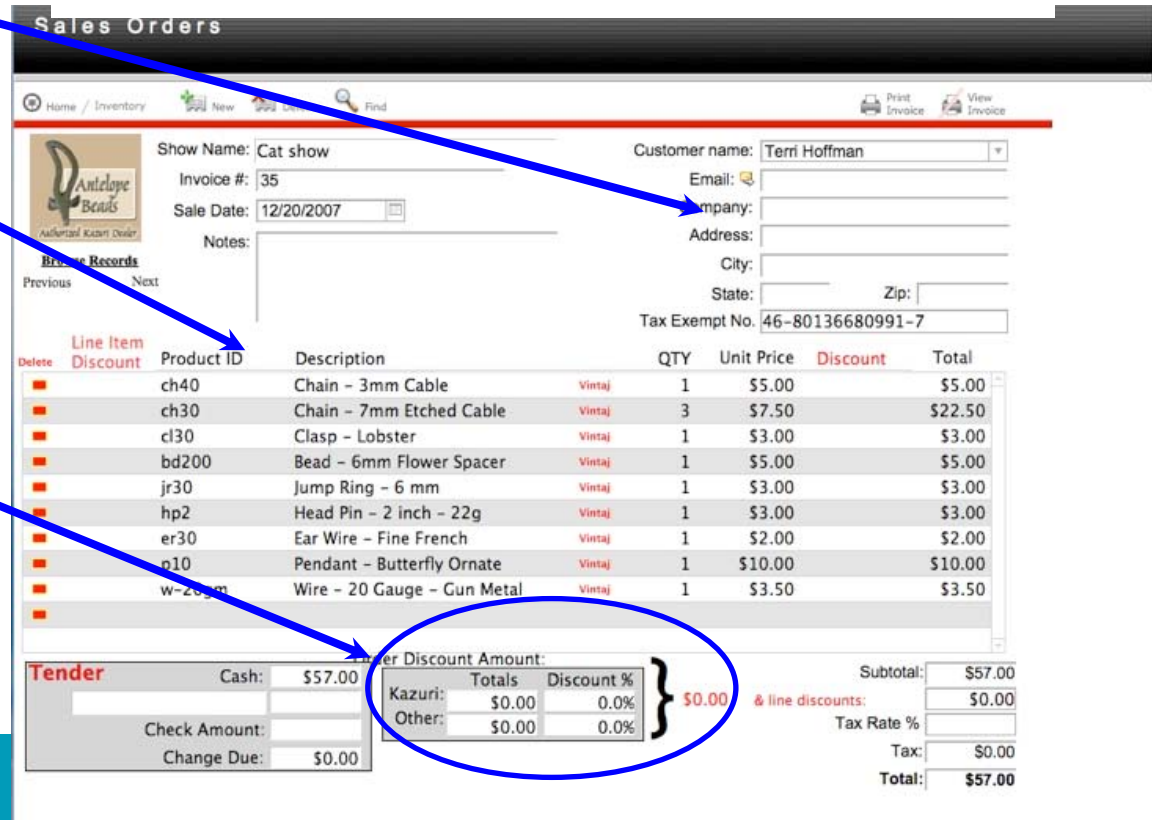
Many tools enable prototypes to be generated from data models (like the domain model)

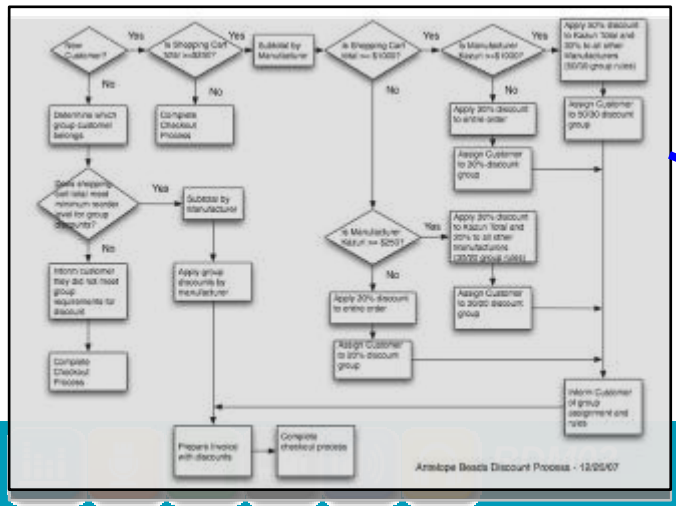
The domain model is easily integrated with other requirements approaches

Example: Handling CRUD behavior









Fully Described: A Very Simple Partial Example

Basic Flow

The use case begins when the *Web Customer* begins browsing the **catalog of product offerings.**

{Browse Catalog}

The system displays the product offerings that are **currently available**, highlighting the **product categories** that the customer has saved in their **profile**.

The *Web Customer* browses through the catalog.

{Select Product}

The *Web Customer* selects a **product** to be purchased by entering the quantity of the product and adding it to their **shopping cart**.

{Validate Stock On Hand}

The system determines that the quantity requested is in stock, then records the **product identifier** and the quantity requested on the **order**, reducing the available quantity in **inventory** by the amount requested.

The Web Customer continues to browse the product catalog and select products to order until they indicate that they are done shopping and wish to process the **order**.

...

Use Cases and User Interface Prototyping

- Visualization helps people to comprehend how the solution will work -- but it can also obscure the “big picture”
- Use cases are useful to capture the usage scenarios, while prototypes are useful for exploring usability issues
- Done in parallel, they can complement each other

Basic Flow

The use case begins when the *Web Customer* begins browsing the **catalog of product offerings**.

(Browse Catalog)

The system displays the product offerings that are **currently available**, highlighting the **product categories** that the customer has saved in their **profile**.

The *Web Customer* browses through the catalog.

(Select Product)

The *Web Customer* selects a **product** to be purchased by entering the quantity of the product and adding it to their **shopping cart**.

(Validate Stock On Hand)

The system determines that the quantity requested is in stock, then records the **product identifier** and the quantity requested on the **order**, reducing the available quantity in **inventory** by the amount requested.

The Web Customer continues to browse the product catalog and select products to order until they indicate that they are done shopping and wish to process the **order**.

(Process Order)

The system prompts the Web Customer for their **billing address** and **shipping address**, which the Web Customer provides.

[Movie-making analogy]

Script

Storyboard

Referencing the Glossary

Use Case – Browse Products & Place Orders - Extract

- The use case begins when the *Web Customer* begins browsing the **catalog of product offerings**.
- The system displays the product offerings that are **currently available**, highlighting the **product categories** that the customer has saved in their **profile**.

…

Glossary Extract

…

catalog of product offerings: an online listing of all products offered for sale.

currently available (products): A product is available if there is a non-zero quantity of that product available for sale.

product category: A product category is a grouping of related products. Products can belong to more than one ctagory

…

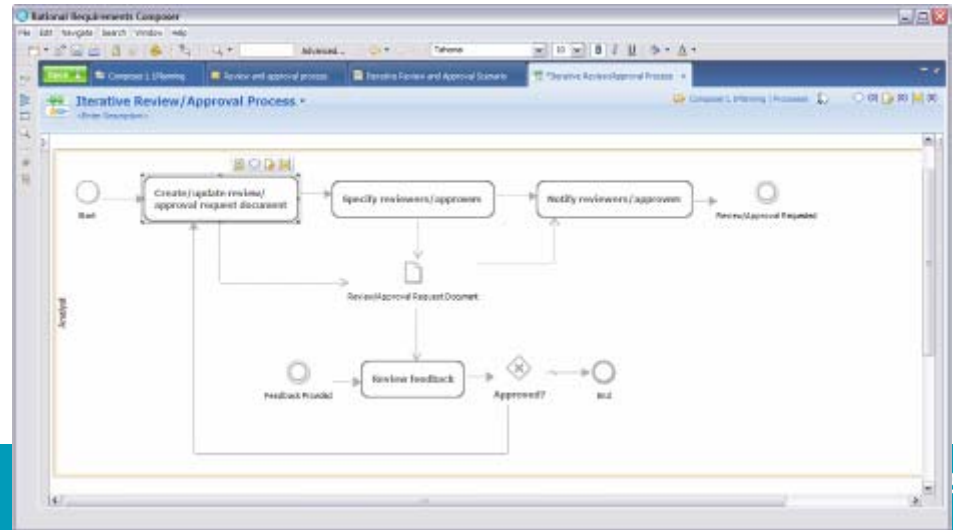
Use **boldface** to highlight Glossary terms. Hyperlinking works well if your tools support this.

Types of Declarative Requirements

- **Cross-cutting requirements**, functional or non-functional, which are requirements that affect many use cases
- **Non-functional requirements** quantify some property of the system or its required behavior but cannot be represented as a behavioral description
- **Constraints** prescribe specific technical solutions, such as use of specific technologies or algorithms
- **Business Rules** describe the operations, definitions and constraints that apply to an organization in achieving its goals

Examples of Business Rules

- Cash and credit card are accepted for payments at shows. Only Credit Card is accepted for online payments.
- Purchased items may be returned within 30 days of purchase if accompanied by receipt.
- Applicable local taxes are computed and added to the invoice at shows.
- Merchandise marked as “on-sale” is not eligible for a discount.



Referencing Business Rules from Use Cases

...

{Browse Catalog}

The system displays the product offerings that are **currently available**, highlighting the **product categories** that the customer has saved in their **profile**.

The *Web Customer* browses through the catalog.

{Select Product}

The *Web Customer* selects a **product** to be purchased by entering the quantity of the product and adding it to their **shopping cart**.

{Validate Stock On Hand}

The system determines that the quantity requested is in stock, then records the **product identifier** and the quantity requested on the **order**, reducing the available quantity in **inventory** by the amount requested.

The Web Customer continues to browse the product catalog and select products to order until they indicate that they are done shopping and wish to process the **order**.

{Process Order}

The system prompts the Web Customer for their **billing address** and **shipping address**, which the Web Customer provides.

The system prompts the Web Customer for their **payment information**, which they provide. The system validates the **address** and **payment information**.

{Apply Discounts}

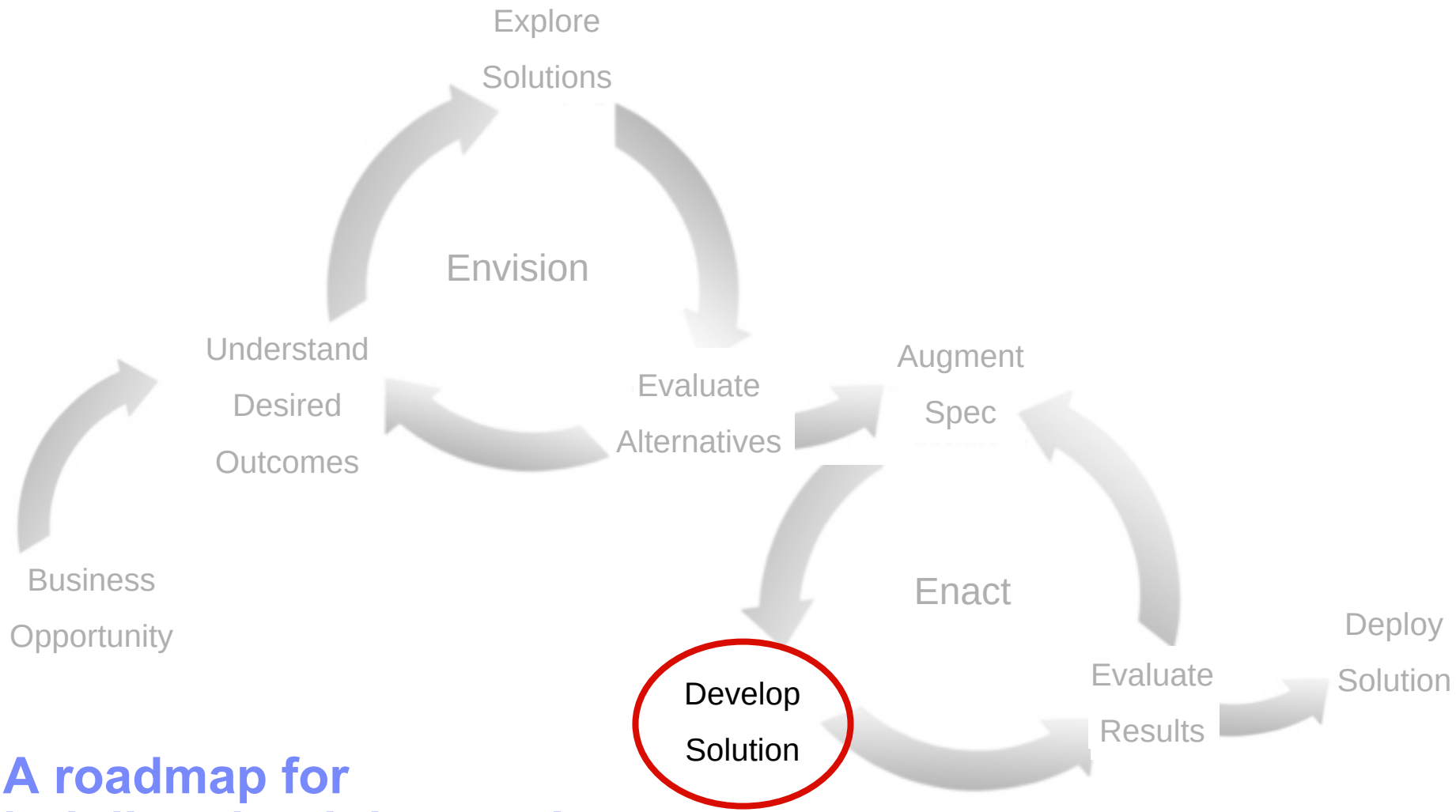
The system applies discounts according to the ***Discount Rules***.

{Apply Payments}

The system charges the amount of the order according to the payment information, and generates a **confirmation notice**. The system also initiates shipment of the order by generating a **pick list**.

The use case then ends.

- Flow steps refer to business rules
- Flow steps give context to the business rules
- Don't repeat Rules in the actual Use Case flow text

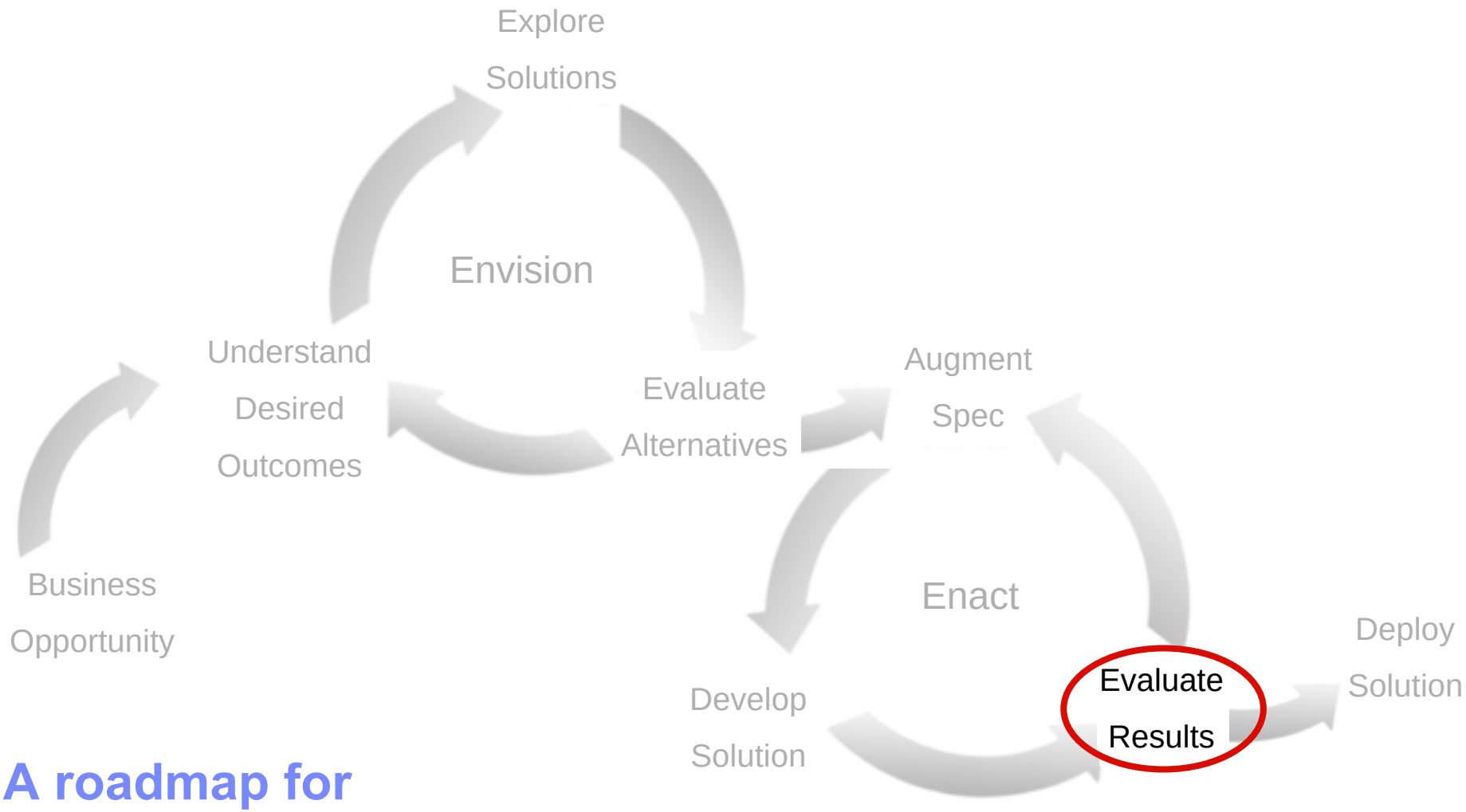


A roadmap for building the right solution

Developing Solutions

- Having visualizations & supporting documentation provides a firm foundation on which to build
- Keeping open communication with the business is important - frequent demonstration and review of working software builds confidence & allows fine-tuning of direction

Frequent review of working software is the surest means of keeping a project on track



A roadmap for building the right solution

Evaluating Results

- Testing is essential for closing the loop - it is a means for evaluating whether the overall project objectives were achieved
- Testing means more than just making sure there are no defects - it also means making sure the solution meets expectations.
  - Comparing results delivered against the desired outcomes will tell you whether you have delivered the right thing
- Involving the business in the effort, continuously throughout the project, ensures that expectations are being met

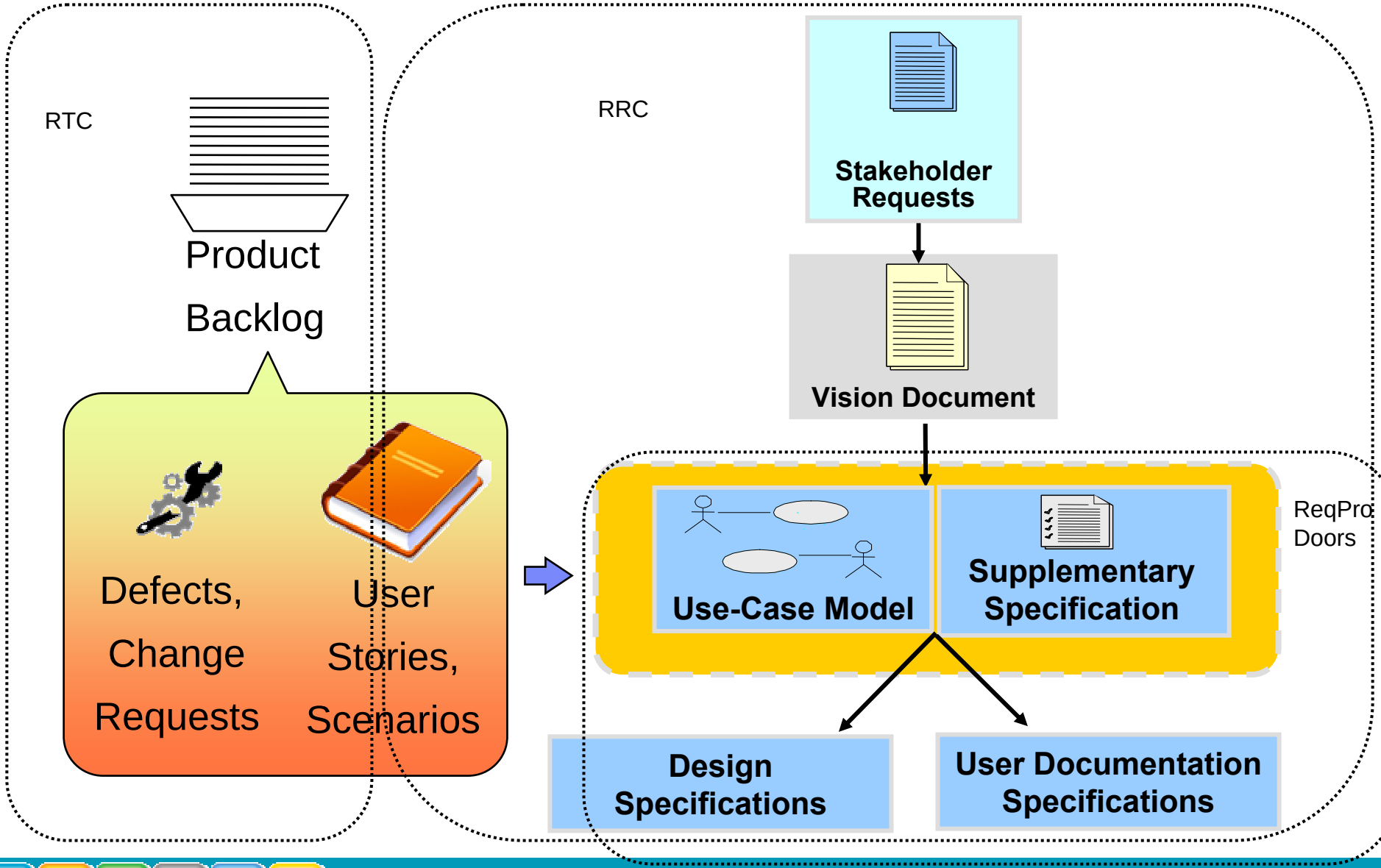
Summary

- Use the simplest possible way to express yourself
  - Start with the Product Backlog
  - Use a Glossary and a Domain Model for information requirements
  - Visualize and prototype the UI
  - Employ use-case specifications for scenarios that have “flow”
- Choose an appropriate level of detail
  - More detail for complex scenarios, higher scrutiny or greater risk
  - Not all requirements need to be documented to the same detail
- Choose an approach that will improve your interactions with stakeholders
  - What do they expect to see?
  - What do they need to see?
  - What is the best way to get feedback?
  - How will you document agreements?

Summary

- Improving results from software projects is mostly about improving communication
- Clarity about desired outcomes is essential, but is often overlooked in the rush to develop solutions
- Visualization of potential solutions is an ideal way to achieve consensus on how the solution will deliver the desired outcomes
- Continuous feedback through the development effort provides a means of assessing progress
- Testing means more than just verifying that there are no defects - it more importantly means closing the loop back to business value

Automation


RTC
Product Backlog
Defects, Change Requests
User Stories, Scenarios
RRC
Stakeholder Requests
Vision Document
Use-Case Model
Supplementary Specification
ReqPro
Doors
Design Specifications
User Documentation Specifications