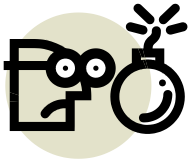
Requirements Analysis

- Classify categories of requirements
- Define the principles of iterative requirements analysis
- Learn about use cases and their elements
- Focusing on the WHAT not the HOW

Requirements Analysis is Hard

- Major causes of project failures
  - Poor user input
  - Incomplete requirements
  - Changing requirements



- Difficulties
  - Complex problems, unknown domains, non-technical customers, communication-intensive
- Essential tools
  - Classification of requirements
  - Use Cases

Classification

- Functional requirements: behavior, features, capabilities
  - "The system reads employee records and prints paychecks"
  - All other reqs are non-functional
- Usability requirements: human factors, help, documentation
  - "The font on the display should be readable from 5 feet"
  - "Do not use colors associated with common forms of color blindness"

Classification

- Reliability requirements: frequency of failure, recoverability
  - "If there is a failure of the external credit card authorization system, …"
- Performance requirements
  - "The server response time is <1 sec for 90% of the accesses"
  - "The system should be capable of processing 1MB of incoming transaction data each second"

Classification

- Supportability requirements:
  - Adaptability, internationalization, maintainability
  - "The system should allow frequent and easy changes in the network configuration"
  - "The system should be capable of incorporating several (unknown) third-party components for tax calculation"
- Implementation requirements
  - "Must use Linux and Java": e.g. decision due to cost, portability, availability of trained personnel

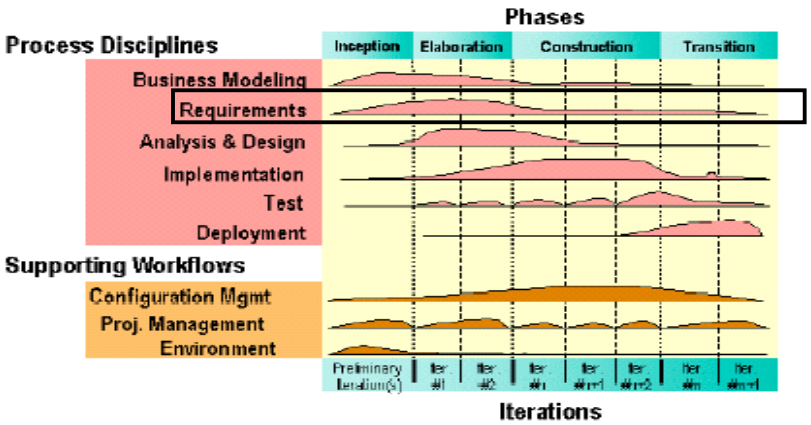
Requirements in Iterative Development

- Often 20%-50% of the original reqs change because of miscommunication or changing business needs
- A key motivation for the iterative model
- The short duration of iterations allows quick adaptation of changed reqs
  - As in how the UP does reqs

Requirements Analysis in the UP

- Major artifacts: Use-Case Model and Supplementary Specification
- Use-Case Model: functional requirements
  - A set of use cases
    - A scenario of a specific use of the system (a text story)
- Supplementary Specification: non-functional requirements
  - Quality reqs: performance, reliability, usability, supportability
  - Others: implementation, etc.

Effort Distribution for Requirements Analysis



source: Rational. software *white paper, May 2002*

Possible Timeline

- Inception, 1 week
  - Use cases are identified and described briefly; 10% are described in detail
- Elaboration, iteration #1, 4 weeks
  - Design a small set of architectural and high-risk requirements; implement and test them
  - End of the iteration: 2 day meeting on the use cases -> 30% are described in detail

Possible Timeline

- Elaboration, iteration #2, 4 weeks
  - Design, implementation, testing of high-risk and architectural requirements
    - 5% of the code is built
  - At the end: detailed description of 50% of the use cases
- Elaboration, iteration #3, 3 weeks
  - More design, implementation, testing; 10% of the code is built
  - 70% of the use cases - fully described

Possible Timeline

- Elaboration, iteration #4, 3 weeks
  - Design, implementation, testing
    - High-risk and architecturally significant aspects should be stabilized
    - Total of 15% of the final system is built
  - 80-90% of the use cases: clarified and written in detail
- Construction
  - Very little work on the use cases
  - Design/coding for the rest of the system

The Role of Use Cases

- Currently, the most widely used approach for capturing requirements
- Central role in the Unified Process
  - Requirements are primarily discovered and recorded through use cases
  - All other activities are driven by the use cases (domain modeling, design, etc.)
- Running example: point-of-sale (POS) system [Larman]

Example

Process Sale use case - A customer arrives at checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system.

Stakeholders and Goals

- **Stakeholders: customer, cashier, company, tax agencies, credit card company**
- **A use case is a story of using the system to fulfill stakeholder goals**
  - **Better than a "laundry list" of features that neither takes a user perspective nor considers user goals**

Some Definitions

- **Actor:** something with behavior – e.g., person, computer system, organization
- **Scenario:** a specific sequence of interactions between the actors and the system under discussion
  - Success scenario or failure scenario
- **Use case: collection of related success and failure scenarios**

Scenarios

Handle Returns use case

- Main success scenario: A customer arrives with items to return. The cashier uses the POS system to record ....
- Alternative Scenarios
  - If they paid by credit but reimbursement to their credit account is rejected, pay by cash
  - If the system detects a failure in the external accounting system, ...
  - (Can you think of other alternatives?)

Black-Box Use Cases

- Do not describe the internal workings of the system
  - Only system responsibilities
  - Goal: specify what the systems should do without deciding how it will do it
- Good: "The system records the sale"
- Bad: "The system writes the sale to a database" or "The system generates SQL INSERT statement for the sale"

Levels of Formality

- Brief: one-paragraph, for the main success scenario
  - *Process Sale* example was brief
- Casual: multiple paragraphs that cover several scenarios
  - *Handle Return* example was casual
- Fully dressed: all steps and variations are written down
  - Developed iteratively during elaboration; the product of requirements analysis

Fully Dressed Use Case

- Primary actor: principal actor that interacts with the system
  - *ProcessSale* - the cashier
- Preconditions and postconditions
- Main success scenario
- Alternative scenarios
- Several other kinds of information
  - Excellent example in [Larman, Section 6.8]

Fully Dressed Use Case - Outline

- Name of use case - start with a verb
- Primary actor - user of system
- Stakeholders - who cares about this use case?
- Preconditions - what has to be true at the start?
- Success guarantee - postcondition on successful completion
- Main success scenario - *happy path* story
- Extensions - alternate scenarios
- Special requirements - non-functional reqs
- Technology List - varying I/O methods, data formats
- Estimate of frequency of occurrence

Scenarios

- **Main success scenario:** the "happy path"
  - No conditions or branching
- **Alternative scenarios** (Extensions)
  - Both successes and failures
  - The Extensions part is typically much longer than the **Main Success Scenario** part

Preconditions and Postconditions

- **Preconditions**: guaranteed to be true before the use case
  - e.g. "Cashier is authenticated"
  - Often the postconditions of a successful scenario from another use case
- **Postconditions**: what must be true after a successful completion of the use case
  - Results of main scenario and successful alternative scenarios

Example: Process Sale, Cash Only

- Primary actor: Cashier
- Precondition: Cashier is authenticated
- Postconditions
  - Sale is logged, accounting and inventory are updated, receipt is generated
- Main success scenario: sequence of interactions with actors

Main Success Scenario

1. Customer arrives with goods
2. Cashier starts a new sale
3. Cashier enters item id
4. System records sale line item and presents description and running total

Repeat 3-4 until Cashier indicates "done"

5. System presents total with taxes. To determine taxes, System uses an external Tax Calculator system

Main Success Scenario (cont)

- Cashier asks Customer for payment
- Cashier enters cash amount tendered, and System presents change due
- System presents receipt
- System logs completed sale and sends sale info to the external Accounting system and to the external Inventory system

Summary

- Several actors: Cashier, Tax Calculator, Accounting, Inventory
  - Customer could be considered an actor, but has no direct interaction with the system
- Black-box point of view
  - "System logs sale" vs. "System writes sale to a file"
- Simplified interactions: no conditions, branching, etc.
- Understandable by non-technical people

Example of Extension

1. Customer arrives with goods
2. Cashier starts a new sale
3. Cashier enters item id, *but it is invalid*
    1. System signals error and rejects entry
    2. Cashier responds to the error
        1. There is a human-readable item ID (UPC code)
            1. Cashier manually enters ID from UPC
            2. System displays price and description
                1. System signals error and Cashier tries alternative method
        2. There is no id but there is a price on the tag
            1. Cashier asks Manager to perform override
            2. Manager performs overrride
            3. Cashier manually enters the price
        3. ……

Another Example: ATM usage

- Who are the actors?
- What is a brief use case?
- What is the main success case?
- What are the possible extensions needed?
- What are some non-functional requirements for this example?