Integrative Systems Engineering
ECE 697 SI
Requirements Engineering

Prof. Michael Zink
Overview

- Requirements
- Failures
- Requirements and life-cycle
- Requirements traceability
- Modeling
- Testing
- Problems and solutions domain
Example: Software System

- Force of change for new products

- Three key factors:
  - *Arbitrary complexity*. Complexity deep inside system’s components. Complexity not limited by hardware.
  - *Instant distribution*. New software can be instantaneously distributed into systems.
  - “*Off-the-Shelf*”. Systems are constructed from ready-made components with corresponding reduction in product development cycle.
Requirements

- “Time to market, with the right product”
- Enables us to agree and visualize “the right product”
- Requirements Engineering:
  1. Defines problem scope
  2. Links all subsequent development information to it.
- Managing development of a solution that is both appropriate and cost-effective.
Requirements

- Basis for every project, defining what stakeholders need from new system.
- Generally expressed in natural language.
- Challenge: Capture need or problem completely and unambiguously without resorting to specialist jargon.
- Drive project activity.
- Provide “navigation chart” and means of steering towards selected destination.
Requirements

- Agreed requirements provide the basis for planning development of a system and accepting it on completion
- Trade-offs: How can impact of change be assessed without adequately detailed model of original system?
- Risk management:
  - Requirements enable management of risk
  - Risk raised against requirements can be tracked
  - Mitigation and fallback plans
Requirements

- Basis for:
  - Project planning
  - Risk management
  - Acceptance testing
  - Trade-off
  - Change control
## Reasons for Project Failures

<table>
<thead>
<tr>
<th>Reason</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incomplete requirements*</td>
<td>13.1%</td>
</tr>
<tr>
<td>Lack of user involvement*</td>
<td>12.4%</td>
</tr>
<tr>
<td>Lack of resources</td>
<td>10.6%</td>
</tr>
<tr>
<td>Unrealistic expectations*</td>
<td>9.9%</td>
</tr>
<tr>
<td>Lack of executive support</td>
<td>9.3%</td>
</tr>
<tr>
<td>Changing requirements/specifications*</td>
<td>8.7%</td>
</tr>
<tr>
<td>Lack of planning</td>
<td>8.1%</td>
</tr>
<tr>
<td>Didn’t need it any longer*</td>
<td>7.5%</td>
</tr>
</tbody>
</table>

*Directly related to requirements*
# Project Success Factors

<table>
<thead>
<tr>
<th>Factor</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>User involvement*</td>
<td>15.9%</td>
</tr>
<tr>
<td>Management support</td>
<td>13.9%</td>
</tr>
<tr>
<td>Clear statement of requirements*</td>
<td>13.0%</td>
</tr>
<tr>
<td>Proper planning</td>
<td>9.6%</td>
</tr>
<tr>
<td>Realistic expectations*</td>
<td>8.2%</td>
</tr>
<tr>
<td>Smaller milestones</td>
<td>7.7%</td>
</tr>
<tr>
<td>Competent staff</td>
<td>7.2%</td>
</tr>
<tr>
<td>Ownership*</td>
<td>5.3%</td>
</tr>
</tbody>
</table>

*Directly related to requirements
Requirements and Quality

- What do you think is a high-quality car?
- Winner of last three FIA World Rally Championships: Citroen
- Not a luxury car, but the right quality of car for the job
- Conformance to requirements: right car for the right job
- Every management decision is a compromise between cost, schedule, and quality.
- Improving requirements means improving the quality of the product.
Requirements and the Lifecycle

- Misconception, requirements engineering is a single phase executed at outset of product development
- E.g., acceptance testing (very last activity in development process):
  - What is a system accepted against?
  - The stakeholder requirements
- Requirements are still in use at final stage
Requirements and the Lifecycle

- Stakeholder Requirements
  - System Requirements
    - Subsystem Requirements
      - Component Requirements
        - Component Test
  - System Test
    - Integration Test
  - Acceptance Test

Testing with respect to requirements
Requirements and the Lifecycle

- Classic V-model used to portray various stages of development
- Development in terms of layers, each corresponding to a development stage
- Basic pattern of requirements use is basically the same at each layer
Requirements and the Lifecycle

Defining results for stakeholders, Validating the product

Defining what system must do, Verifying the system

Optimizing cost-benefits, Qualifying requirements

Allocating requirements, Qualifying components

Acceptance Test

System Test

Component Test

System Requirements

Subsystem Requirements

Component Requirements

Stakeholder Requirements
Requirements and the Lifecycle

- Requirements can act as means of communication between projects:
  - Maximize reuse of artefacts across projects
  - Manages families of similar projects
  - Use program management to coordinate activities
  - Optimize process by learning from experience of other projects
- Stakeholder requirements can provide concise, non-technical description to senior management
- System requirements can form excellent technical summary of project
Requirements and the Lifecycle

- Stakeholder Requirements
- System Requirements
- Subsystem Requirements
- Component Requirements

Inform the enterprise
- Learning from the enterprise

Acceptance Test
- System Test
- Integration Test
- Component Test

Inform the enterprise
- Learning from the enterprise

Inform the enterprise
- Learning from the enterprise

U Mass Amherst
Requirements and the Lifecycle

- Requirements need to be maintained
- Change design without changing requirements is problematic
- Requirements engineering connects strongly with change management:
  - Accept or reject change
  - Negotiate the cost of change with customers/suppliers
  - Organize the redevelopment work
- Concept that enables impact analysis is requirements traceability
Requirements and the Lifecycle

Using traceability and impact analysis to manage change

Stakeholder Requirements
System Requirements
Subsystem Requirements
Component Requirements
Acceptance Test
System Test
Integration Test
Component Test

Using traceability and impact analysis to manage change
Requirements and the Lifecycle

- Project manager’s and systems engineer’s ability to control project enhanced by good requirements
  - No means of gauging how well project is going
  - Or if it is going into right direction
- No basis against which change can be judged
- Requirements give right view of project
- Requirements essential to healthy system development
- Beginning to end and top to bottom
- Voice of customer becomes clear line of communication
Requirements Traceability

- Traceability:
  - Understanding how high-level requirements – objectives, goals, needs – are transformed in low-level requirements

- Primarily concerned with relationship between layers of information

- In business context:
  - Business vision is interpreted as
  - Business objectives are implemented as
  - Business organizations and processes

- In engineering context:
  - Stakeholder requirements are met by
  - System requirements are partitioned into
  - Subsystems are implemented as
  - Components
Requirements Traceability

- Using traceability can contribute to:
  - Greater confidence in meeting objectives
    Greater reflection on how objectives are satisfied
  - Ability to assess the impact of change
    Various forms of impact analysis
  - Improved accountability of subordinate organizations
    Clarity of how suppliers contribute to the whole
  - Ability to track progress
    Traceability allows precise measure of progress in early stages
  - Ability to balance cost against benefit
    Relating product components to requirements
Requirements Traceability

- Relationships are usually many-to-many
- May be linked to higher- and lower-level requirements
- Link statements on one layer with the ones in another
- These links are somewhat like bidirectional hyperlinks in web pages
- Direction of arrows follows particular convention: Information traces back to information it responds to:
  - Chronological order in which information is created
  - Corresponds to access rights due to ownership
Requirements Traceability

- Stakeholder Requirements
- System Requirements
- Subsystem Requirements
- Component Requirements
- Acceptance Test plan
- System Test plan
- Integration Test plan
- Component Test plan
## Requirements Traceability

<table>
<thead>
<tr>
<th>Type of Analysis</th>
<th>Description</th>
<th>Processes Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impact analysis</td>
<td>Following incoming links, in answer to the question “What if this was to change?”</td>
<td>Change management</td>
</tr>
<tr>
<td>Derivation Analysis</td>
<td>Following outgoing links, in answer to the question “Why is this here?”</td>
<td>Cost-benefit analysis</td>
</tr>
<tr>
<td>Coverage Analysis</td>
<td>Counting statements that have links, in answer to the question “Have I covered everything?”</td>
<td>General engineering Management reporting</td>
</tr>
</tbody>
</table>
Requirements Traceability: Impact Analysis

- Stakeholder Requirements
- System Requirements
- Subsystem Requirements
- Component Requirements
- Acceptance Test plan
- System Test plan
- Integration Test plan
- Component Test plan

Impact Analysis
Derivation Analysis
Requirements Traceability: Coverage Analysis

Are all requirements covered by the layer below?

Are all requirements covered by tests?

Stakeholder Requirements

System Requirements

Subsystem Requirements

Component Requirements

Acceptance Test plan

System Test plan

Integration Test plan

Component Test plan

Are all requirements covered by tests?
Requirements Traceability: Coverage Analysis

- Coverage analysis can also be used to measure progress
  - How far have Systems Engineers got in responding to stakeholder requirements
  - Systems Engineers write systems requirements which can be traced back to stakeholder requirements.
- Creating traceability adds little overhead if it’s done in parallel to systems requirements development
- Very useful management tool in early stages of development
Important to understand relationship between requirements management and system modeling
- Mutually supportive activities that should not be equated
- Relationship almost like a sandwich:
  - Requirements management: *Bread & Butter*
  - Modeling: *Filling*

There is no such thing as *requirements modeling*

Assists engineer in decomposing requirements
Requirements and Modeling

- Particular model never says everything about system – if it did, it wouldn’t be a model!
- Model is abstraction of system that focuses on some aspect to the exclusion of others
- Ignoring details that are not relevant to particular model
- Smaller amounts of related information can be collected, processed, organized, and analyzed
- Modeling allows zooming-in where large amounts of complex information has to be managed
- Aids in maintaining system-wide grasp through focusing on small amount of information at a time
Requirements and Modeling

- Models assist requirements engineer in analyzing the requirements at a particular level so as to:
  - Communicate with the customer and improve mutual understanding of the system to be developed
  - Analyze the system to ascertain the presence of desired emergent properties
  - Determine how to satisfy the requirements by deriving new requirements at the layer below

- Functional model may be used to derive system requirements from the stakeholder requirements
Requirements and Modeling

Requirements Layer

Modeling Layer

Requirements Layer

Modeling Layer

Requirements Layer

Modeling Layer

Requirements Layer

Statement of need

Usage modeling

Stakeholder requirements

Functional modeling

System requirements

Performance modeling

Subsystem requirements
Models have many facets and may include queuing theory, wind tunnels, etc.

Nature of model varies from application to application:

- Modeling of timetable appropriate for design of railway system
- In aircraft design the modeling of aerodynamics might be more appropriate
- What about aerodynamics for high-speed trains?
- Message sequence charts for communication system ...
- ... but data-focused modeling for data-rich application
Requirements and Testing

- Testing is closely related to requirements at every level
- Allows defects in a system to be detected (defect is a departure from requirements)
- Testing activities include (in addition to classical tests of components, subsystems and systems):
  - Reviews
  - Inspection
  - Analysis through modeling

- Example: TTCN
Requirements and Testing

- Testing should begin as early as possible
- Earliest tests take place during design of system and include:
  - Requirements review
  - Design inspection
  - Various forms of analysis carried out on system models
Requirements and Testing

- Stakeholder Requirements
  - System Requirements
    - Subsystem Requirements
      - Component Requirements
    - Component tests
  - System tests
  - Rig tests
  - System tests
  - Trials

- Acceptance Test
  - Integration Test
    - Component Test
Problem and Solution Domain

- Clear distinction should be made between problem domain and solution domain

- Problem domain:
  - Statement of need
  - Usage modeling
  - Stakeholder requirements

- Solution domain:
  - System requirements
  - System component requirements
  - Subsystem component requirements
## Problem and Solution Domain

<table>
<thead>
<tr>
<th>Requirements Layer</th>
<th>Domain Layer</th>
<th>View</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholder requirements</td>
<td>Problem domain</td>
<td>Stakeholder’s view</td>
<td>State <em>what</em> the stakeholders want to achieve through use of the system. Avoid reference to any particular solution.</td>
</tr>
<tr>
<td>System requirements</td>
<td>Solution domain</td>
<td>Analyst’s view</td>
<td>State abstractly <em>how</em> system will meet the stakeholder requirements. Avoid reference to any particular design.</td>
</tr>
<tr>
<td>Architectural design</td>
<td>Solution domain</td>
<td>Designer’s view</td>
<td>State <em>how</em> the specific design will meet the system requirements.</td>
</tr>
</tbody>
</table>
Problem and Solution Domain

- Initial statement of capability should state no more than necessary to define problem. Avoid reference to particular solution.
- Gives freedom to Systems Engineer since task is to devise best solution without preconceived ideas.
- Modeling assists in derivation of next layer of requirements.
- Early modeling should focus on enclosing system.
Problem and Solution Domain

- Example: Traffic controls system
  - Stakeholder may express problem in terms of maximizing traffic flow while minimizing risk of accidents and minimizing cost of maintenance
  - Systems Engineer may consider a variety of solutions: Traffic lights, roundabouts, bridges as approach that best solves problem (constraint of development, maintenance cost)
  - Designers then start designing selected solution
Problem and Solution Domain

- Without clear distinction between problem solution following may result:
  - Lack of understanding of the real problem
  - Inability to scope the system and understand which functions to include
  - Domination of debate about the system by the developers and suppliers because only descriptions of the system are expressed in terms of solutions
  - Inability to find optimal solutions due to lack of design freedom