Non-Functional Requirements (NFRs) & Requirements Prioritization

Department of Software Engineering,
College of Information & Communication Technology,
Can Tho University

Tran Van Hoang

What are Non-functional Requirements?

- **IEEE definition**
  - **Non-functional** requirements in software system engineering, a software requirement that describes not what the software will do, but how the software will do it, for example, software performance requirements, software external interface requirements, design constraints, and software quality attributes.
  - Nonfunctional requirements are difficult to test.
  - Therefore, they are usually evaluated subjectively.

What are Non-functional Requirements?

- **Functional vs. Non-Functional**
  - **Functional** requirements describe what the system should do.
    - functions that can be captured in use cases.
    - behaviors that can be analyzed by drawing sequence diagrams, state charts, etc.

What are Non-functional Requirements?

- **Functional vs. Non-Functional**
  - **Non-functional** requirements are global constraints on a software system.
    - e.g. development costs, operational costs, performance, reliability.
    - Maintainability, portability, robustness etc.
    - Often known as software qualities, or just the "**ilities**".
    - Usually cannot be implemented in a single module of a program.

What are Non-functional Requirements?

- NFRs for a computer game.
  1. Can run on any operating system.
  2. The password should be encrypted for safety.
  3. Software can run on any internet explorer, firefox, google chrome.
  4. Must be free of defects.
  5. Every video must have an unique identification.
  6. All requests must return a response in < 2 seconds.
  7. …

What are Non-functional Requirements?

- **The challenge of NFRs**
  - Hard to model.
  - Usually stated informally, and so are:
    - difficult to implement during development
    - difficult to evaluate for the customer prior to delivery
  - Hard to make them measurable requirements
    - We'd like to state them in a way that we can measure how well they've been met.

Example NFRs

- Interface requirements
  - How will the new system interface with its environment?
    - User interfaces
    - Interfaces with other systems
- Performance requirements
  - Time/space bounds
    - response time, throughput and available storage space
    - e.g. ”the system must handle 1,000 transactions per second”
    - Query response time must be fast. (not measurable)

Example NFRs

- Reliability
  - The availability of components.
  - Integrity of information maintained and supplied to the system.
    - e.g. "system must have less than 1hr downtime per three months“.
    - e.g. The system must never crash. (not feasible)
- Security
  - One or more requirements about protection of your system and its data.
    - E.g. permissible information flows, or who can do what

Example NFRs

- Portability
  - It must be possible to port the system to the Linux platform with < 12 person-months of effort.
- Economic requirements
  - e.g. restrictions on immediate and/or long-term costs.
- Operating requirements
  - Physical constraints (size, weight)
  - Personnel availability & skill level
  - Accessibility for maintenance
  - Environmental conditions

Example NFRs

- Lifecycle requirements
  - "Future-proofing"
    - Maintainability
    - Enhanceability
  - Limits on development
    - E.g. development time limitations
    - Resource availability
    - Methodological standards

Requirements Prioritization

- Why Prioritization is needed?
- Cost-Value Approach
  - Sorting Requirements by cost/value
  - Estimating Relative Costs/Values using AHP

Why prioritize requirements?

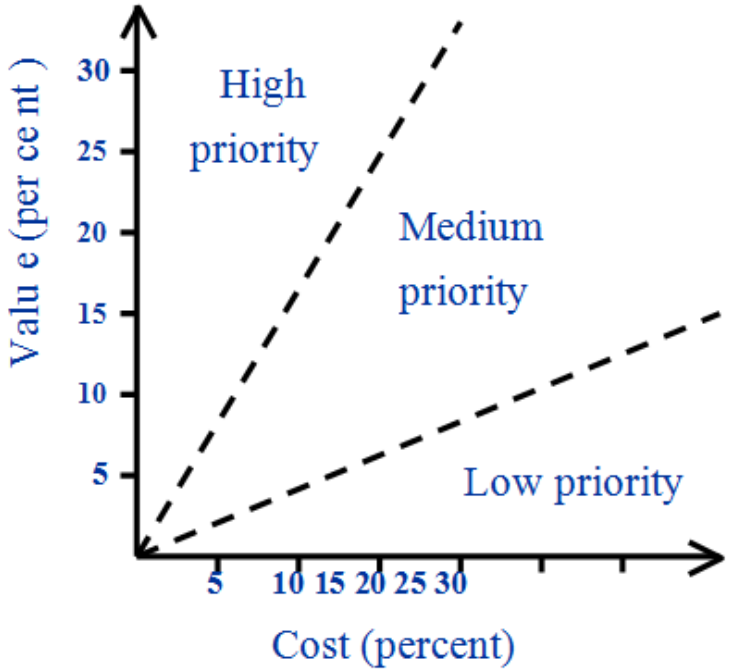
- When customer expectations are high and timelines are short, you need to make sure the product delivers the most critical or valuable functionality as early as possible.
- Prioritization is a way to deal with competing demands for limited resources.
- Prioritization is a critical strategy for agile projects.

Basics of Prioritization

- **Need to select what to implement**
  - Customers (usually) ask for way too much.
  - Balance time-to-market with amount of functionality.
  - Decide which features go into the next release.
- **For each requirement/feature, ask:**
  - How important is this to the customer?
  - How much will it cost to implement?
  - How risky will it be to attempt to build it?

A Cost-Value Approach

- **Calculate return on investment (ROI)**
  - Assess each requirement's importance to the project as a whole.
  - Assess the relative cost of each requirement.
  - Compute the cost-value trade-off:



Estimating Cost & Value

- **Two approaches:**
  - Absolute scale (e.g. dollar values)
    - Requires much domain experience.
  - Relative values (e.g. less/more; a little, somewhat, very)
    - Much easier to elicit.
    - Prioritization becomes a sorting problem .

Estimating Cost & Value

- **Comparison Process - options**
  - Basic sorting - for every pair of requirements (i, j), ask if i>j?
    - E.g. bubblesort - start in random order, and swap each pair if out of order.
    - requires n*(n-1)/2 comparisons
  - Construct a Binary Sort Tree
    - Requires O(nlogn) comparisons

Some complications

- Hard to quantify differences
  - Easier to say "*x is more important than y*"…
  - …than to estimate by how much
- Not all requirements comparable
  - E.g. different level of abstraction
- Requirements may not be independent
  - No point selecting between X and Y if they are mutually dependent

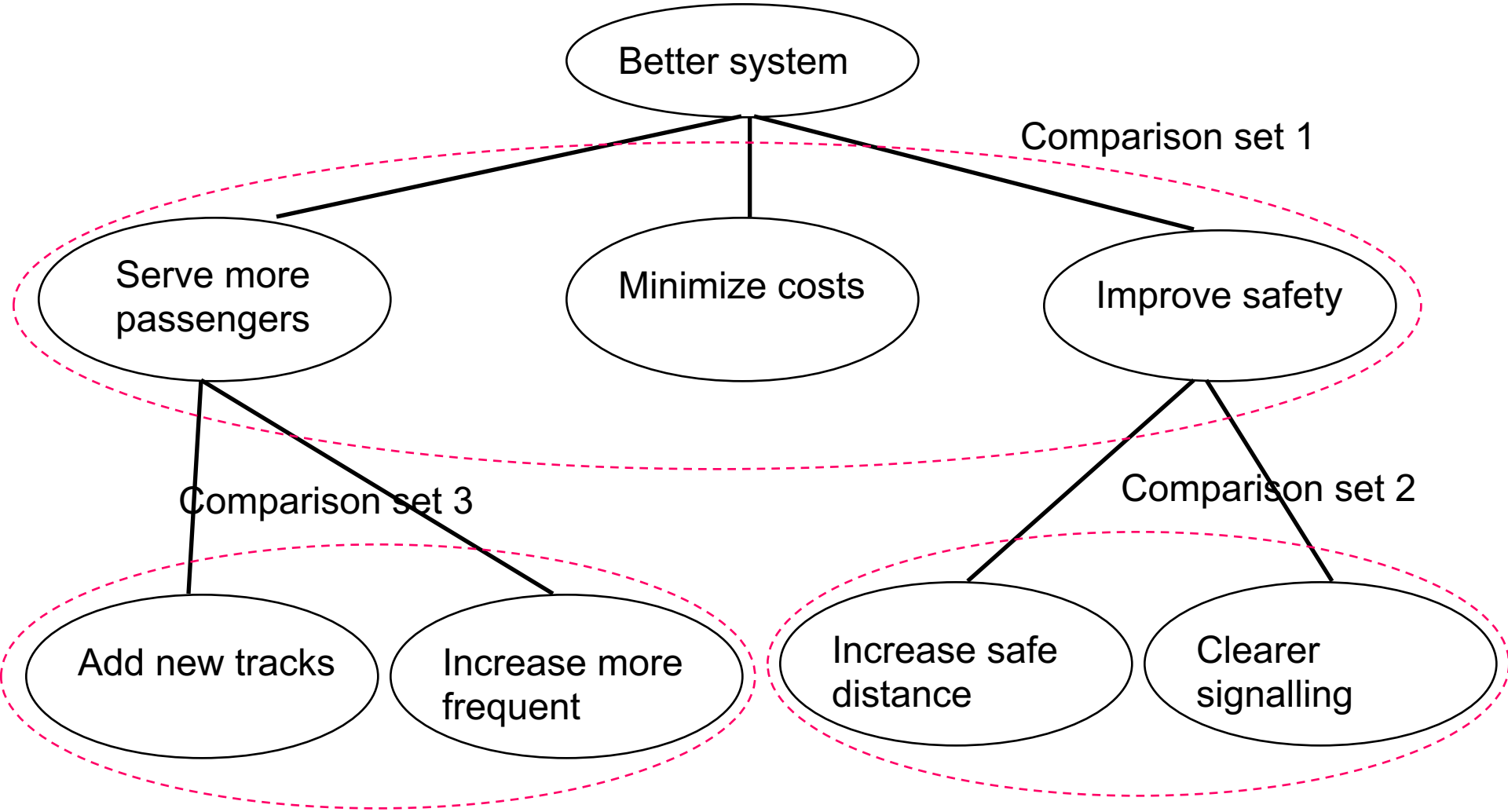
Some complications

- Stakeholders may not be consistent
  - E.g. If X > Y, and Y > Z, then presumably X > Z?
- Stakeholders might not agree
  - Different cost/value assessments for different types of stakeholder

Hierarchical Prioritization

- Group Requirements into a hierarchy
  - E.g. A goal tree
  - E.g. A NFR tree
- Only make comparisons between branches of a single node:

Hierarchical Prioritization



Analytic Hierarchy Process (AHP)

- Create n x n matrix (for n requirements)
  - For cell (x,y) in the matrix enter:
    - 1 - if x and y are of equal value
    - 3 - if x is slightly more preferred than y
    - 5 - if x is strongly more preferred than y
    - 7 - if x is very strongly more preferred than y
    - 9 - if x is extremely more preferred than y
    - (use the intermediate values, 2, 4, 6, 8 if compromise needed)
  - …and for (y,x) enter the reciprocal

Analytic Hierarchy Process (AHP)

- Estimate the eigenvalues:
  - E.g. “averaging over normalized columns”
    1. Calculate the sum of each column.
    2. Divide each element in the matrix by the sum of it’s column.
    3. Calculate the sum of each row.
    4. Divide each row sum by the number of rows.

AHP example - estimating costs

	Req1	Req2	Req3	Req4
Req1	1	1/3	2	4
Req2	3	1	5	3
Req3	1/2	1/5	1	1/3
Req4	1/4	1/3	3	1

Normalise columns

	Req1	Req2	Req3	Req4
Req1	0.21	0.18	0.18	0.48
Req2	0.63	0.54	0.45	0.36
Req3	0.11	0.11	0.09	0.04
Req4	0.05	0.18	0.27	0.12

Sum the rows

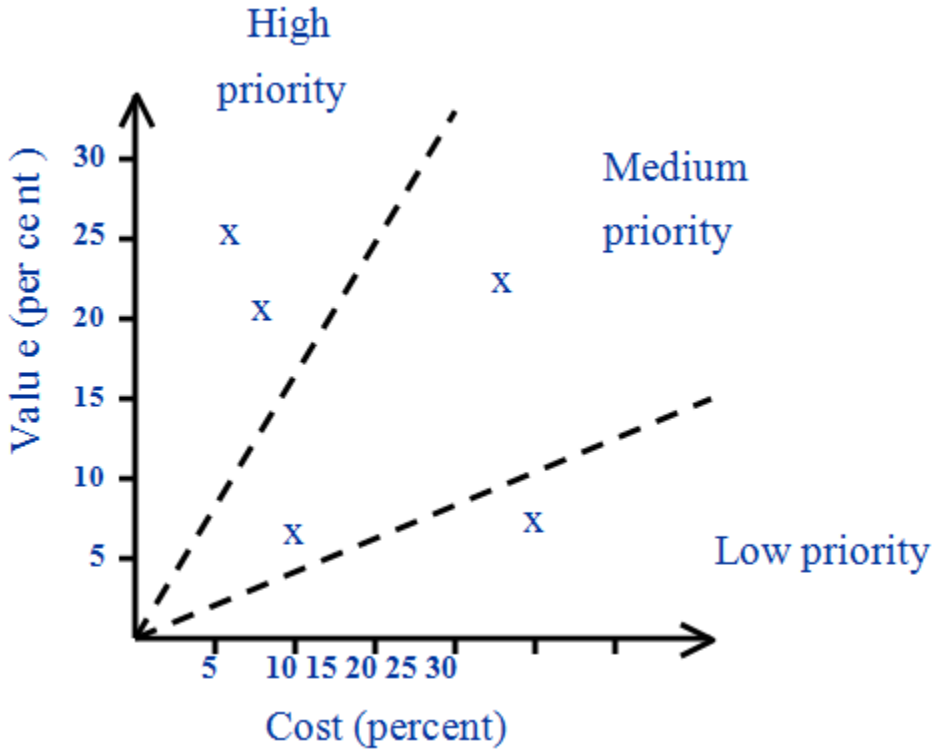
sum	sum/4
1.05	0.26
1.98	0.50
0.34	0.09
0.62	0.16

Result

Req1 - 26% of the cost
Req2 - 50% of the cost
Req3 - 9% of the cost
Req4 - 16% of the cost

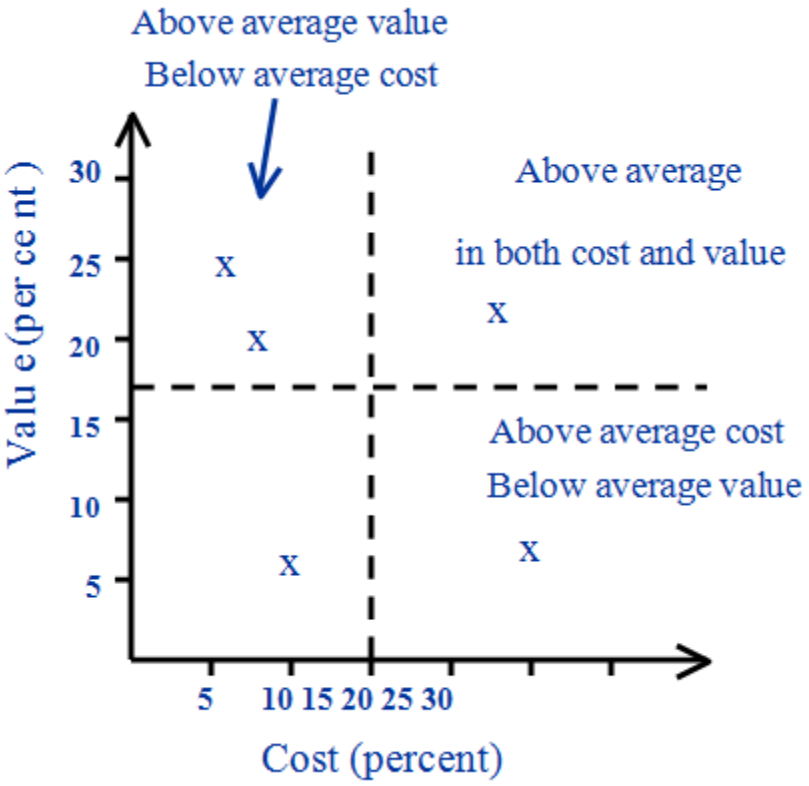
Plot ROI graph

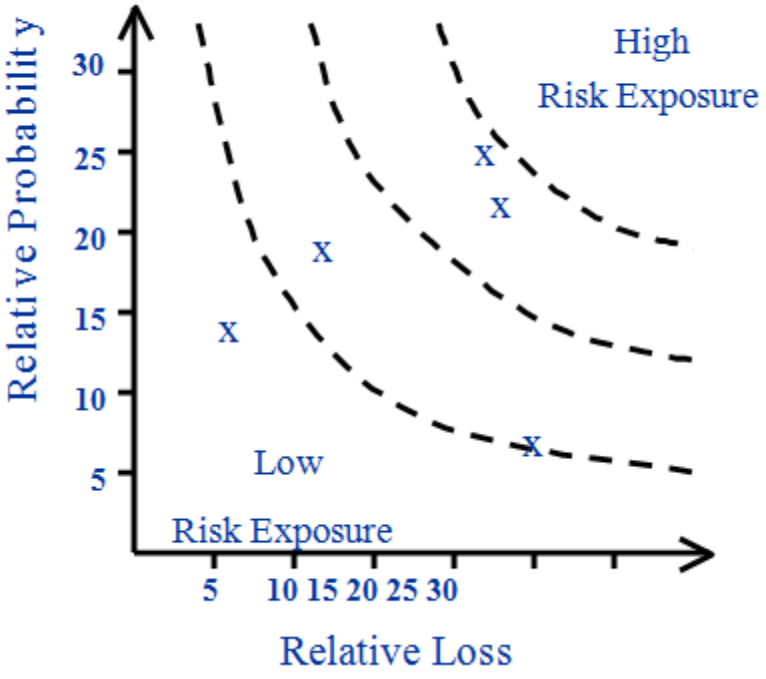
- **Do AHP process twice:**
  - Once to estimate relative value
  - Once to estimate relative cost
- **Use results to calculate ROI ratio:**



Other selection criteria

- **ROI ratio is not the only way to group requirements**





Other techniques

- **In or out**
  - Very simple
  - A group of stakeholders work down a list of requirements and make a binary decision: is it in, or is it out?
  - Keep referring to the project's business objectives to make this judgment.
- **$100**
- **MoSCoW**
- **Three-level scale**

Karl Wiegers and Joy Beatty, Software Requirements, Chapter 16. pp.313-327

Main references

[1] Prof Steve Easterbrook, lecture notes, University of Toronto, Canada.

[2] Karl Wiegers and Joy Beatty, Software Requirements, Chapter 16. pp.313-327

Q&A