CS5351 Software engineering 2024/25 Semester A

Requirements Engineering

Dr W.K. Chan

Department of Computer Science

Email: **`wkchan@cityu.edu.hk`**

Website: **`http://www.cs.cityu.edu.hk/~wkchan`**

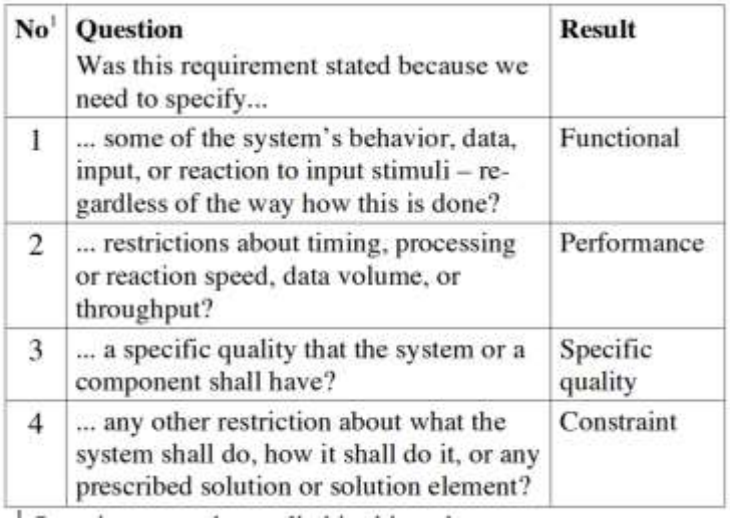
Example

- "As a customer, I want to buy a PS5 from the website."
  - This is a **functional** and **verifiable** requirement.
- Development Team's **Expectation**: Delivering the app satisfying the requirement will make the customer happy.
- Sufficient? Many other concerns …
  - [Efficiency Issue] How about the customer needs to wait 30 minutes after each click or the input to each textbox on the website to complete a purchase of an item?
  - [Scalability Issue] How about the website only allows exactly one customer to buy exactly one item at any time?
  - …
- We must know the non-functional part of the requirements

Requirements Engineering (RE)

- RE is a process to ***find out*** and ***structure*** the **functional** and **non-functional** **requirements** of the software to be built.

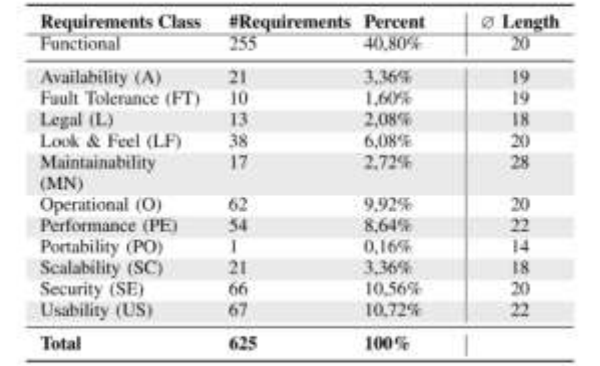
Table 3. Classification rules [6]



No[1]	Question Was this requirement stated because we need to specify...	Result
1	... some of the system's behavior, data, input, or reaction to input stimuli – regardless of the way how this is done?	Functional
2	... restrictions about timing, processing or reaction speed, data volume, or throughput?	Performance
3	... a specific quality that the system or a component shall have?	Specific quality
4	... any other restriction about what the system shall do, how it shall do it, or any prescribed solution or solution element?	Constraint

Example dataset [7]

TABLE I: Overview of the "Quality attributes (NFR)" dataset.



Requirements Class	#Requirements	Percent	⌀ Length
Functional	255	40,80%	20
Availability (A)	21	3,36%	19
Fault Tolerance (FT)	10	1,60%	19
Legal (L)	13	2,08%	18
Look & Feel (LF)	38	6,08%	20
Maintainability (MN)	17	2,72%	28
Operational (O)	62	9,92%	20
Performance (PE)	54	8,64%	22
Portability (PO)	1	0,16%	14
Scalability (SC)	21	3,36%	18
Security (SE)	66	10,56%	20
Usability (US)	67	10,72%	22
Total	**625**	**100%**	

In many cases, a high proportion of all requirements are non-functional requirements

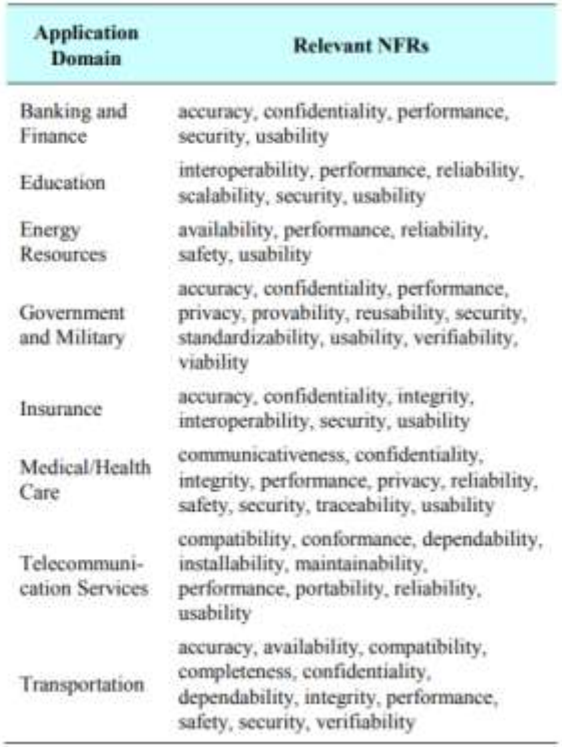
Many faces of non-functional requirements

- Be careful!
- Different people in different domains may use different terminologies
- Example “words” from users to mean non-functional requirements :
  - System properties/characteristics/constraints
  - Quality attributes, non-behavioral requirements, concerns, goals, extra-functional requirements, quality requirements, and system attributes

Many faces of non-functional requirements

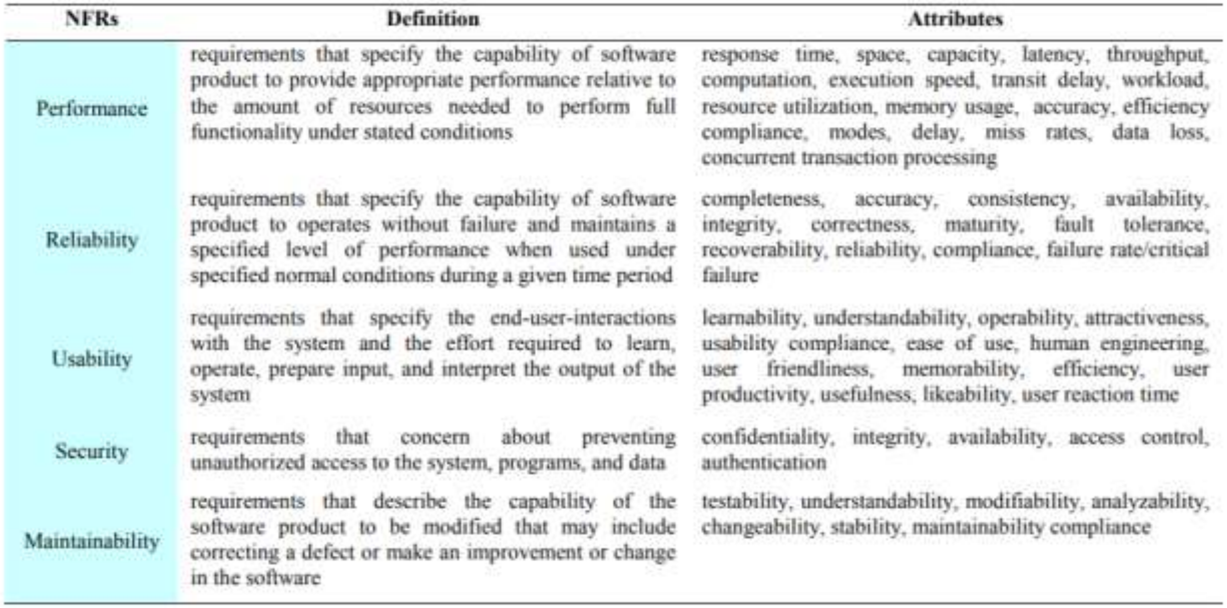
- Different application domains have different sets of major quality attributes.

Table 3 - Application Domains and Relevant NFRs



Application Domain	Relevant NFRs
Banking and Finance	accuracy, confidentiality, performance, security, usability
Education	interoperability, performance, reliability, scalability, security, usability
Energy Resources	availability, performance, reliability, safety, usability
Government and Military	accuracy, confidentiality, performance, privacy, provability, reusability, security, standardizability, usability, verifiability, viability
Insurance	accuracy, confidentiality, integrity, interoperability, security, usability
Medical/Health Care	communicativeness, confidentiality, integrity, performance, privacy, reliability, safety, security, traceability, usability
Telecommunication Services	compatibility, conformance, dependability, installability, maintainability, performance, portability, reliability, usability
Transportation	accuracy, availability, compatibility, completeness, confidentiality, dependability, integrity, performance, safety, security, verifiability

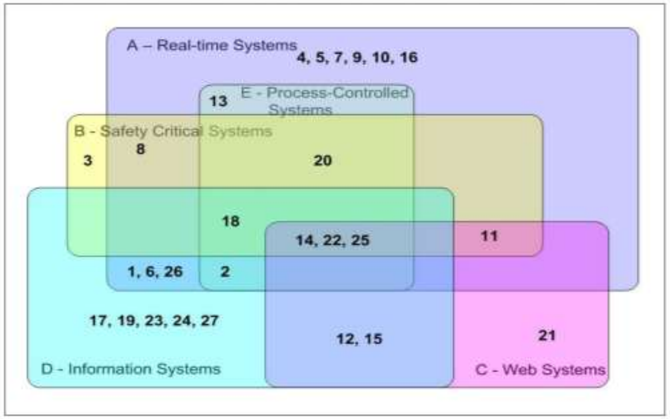
Table 2 - The Most Commonly Considered NFRs



NFRs	Definition	Attributes
Performance	requirements that specify the capability of software product to provide appropriate performance relative to the amount of resources needed to perform full functionality under stated conditions	response time, space, capacity, latency, throughput, computation, execution speed, transit delay, workload, resource utilization, memory usage, accuracy, efficiency compliance, modes, delay, miss rates, data loss, concurrent transaction processing
Reliability	requirements that specify the capability of software product to operates without failure and maintains a specified level of performance when used under specified normal conditions during a given time period	completeness, accuracy, consistency, availability, integrity, correctness, maturity, fault tolerance, recoverability, reliability, compliance, failure rate/critical failure
Usability	requirements that specify the end-user-interactions with the system and the effort required to learn, operate, prepare input, and interpret the output of the system	learnability, understandability, operability, attractiveness, usability compliance, ease of use, human engineering, user friendliness, memorability, efficiency, user productivity, usefulness, likeability, user reaction time
Security	requirements that concern about preventing unauthorized access to the system, programs, and data	confidentiality, integrity, availability, access control, authentication
Maintainability	requirements that describe the capability of the software product to be modified that may include correcting a defect or make an improvement or change in the software	testability, understandability, modifiability, analyzability, changeability, stability, maintainability compliance

Many faces of non-functional requirements

- Even in the same application domain, different types of applications have different sets of non-functional requirements



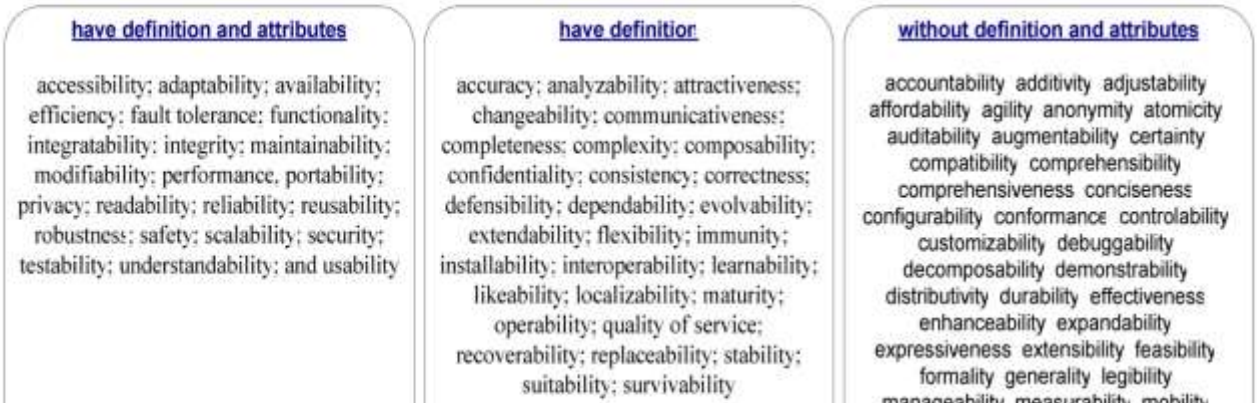
Legend:

1	Accuracy	10	Installability	19	Reusability
2	Availability	11	Integrity	20	Safety
3	Communicativeness	12	Interoperability	21	Scalability
4	Compatibility	13	Maintainability	22	Security
5	Completeness	14	Performance	23	Standardizability
6	Confidentiality	15	Privacy	24	Traceability
7	Conformance	16	Portability	25	Usability
8	Dependability	17	Provability	26	Verifiability
9	Extensibility	18	Reliability	27	Viability

Figure 5 - Type of Systems and Relevant NFRs

Many faces of non-functional requirements

- Some quality attributes are undefined or vaguely known.
- If a quality attribute **cannot** be measured or **cannot** be verified, we **cannot** know whether we cannot predict how well our software meets them
  - For such quality attributes, we can only rely on user evaluations on the software



have definition and attributes

accessibility; adaptability; availability; efficiency; fault tolerance; functionality; integratability; integrity; maintainability; modifiability; performance, portability; privacy; readability; reliability; reusability; robustness; safety; scalability; security; testability; understandability; and usability

have definition

accuracy; analyzability; attractiveness; changeability; communicativeness; completeness; complexity; composability; confidentiality; consistency; correctness; defensibility; dependability; evolvability; extendability; flexibility; immunity; installability; interoperability; learnability; likeability; localizability; maturity; operability; quality of service; recoverability; replaceability; stability; suitability; survivability

without definition and attributes

accountability additivity adjustability affordability agility anonymity atomicity auditability augmentability certainty compatibility comprehensibility comprehensiveness conciseness configurability conformance controlability customizability debuggability decomposability demonstrability distributivity durability effectiveness enhanceability expandability expressiveness extensibility feasibility formality generality legibility manageability measurability mobility

We need to Handle Multi-Level Concerns

- specific to the application domain
  - => meet the general requirement of the industry sector
- specific to the type of software application
  - => meet the general requirement for the type of software
- the specific to the application
  - => meet the unique requirements of the current application
- In all levels, consider non-functional requirements

- If not, our application will not be used easily in the application environment

Requirements Engineering

Requirements Engineering (RE)

- **is the first step** in finding a solution for a data processing problem
- the *results* of requirements engineering is a **requirements specification**
- requirements specification is a
  - contract for the customer
  - starting point for design

Avoid Common Mistakes in RE [https://ieeexplore.ieee.org/document/1695257]

- **noise**: does not add relevant information
- **silence**: feature in the problem not mentioned in the specification
- **over-specification**: talk about the solution rather than the problem
- **contradictions**: inconsistent description
- **ambiguity**: unclear
- **forward references**: especially cumbersome in long documents
- **wishful thinking**: features that cannot realistically be realized

Natural language specs are dangerous

e.g., “All users have the same control field”

- The same value in the control field?
- The same format of the control field?
- There is one (1) control field for all users?

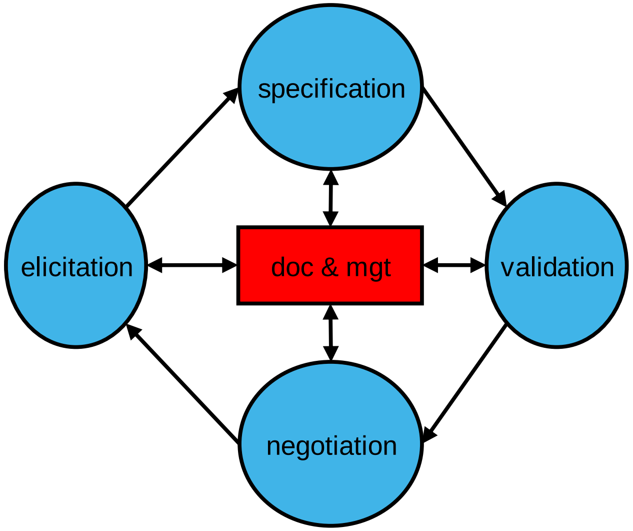
- We should *validate* the *documented* requirements to avoid misunderstanding. It leads to the RE process (see the next slide).

Requirements engineering, main steps

Four major steps in an RE process

1. understanding the problem: **elicitation**
2. describing the problem: **specification**
3. agreeing upon the nature of the problem: **validation**
4. agreeing upon the boundaries of the problem: **negotiation**

This is an iterative process



Conceptual modeling

- We **explicitly** model the scope of the requirements as a domain model

- Making a model explicit requires solving two problems:
  - **Analysis**: to find and clear ambiguity: Because of the unspoken assumptions, different languages /terminologies being spoken, incomplete codification of the problem domain
  - **Negotiation**: to resolve ambiguity due to stakeholders with different goals (e.g., workers against management).
    - As an analyst, we must participate in <u>shaping</u> the domain model

- Having an explicit model reduces later surprises.

How we study the world around us

- people have a set of assumptions about a topic they study
- this set of assumptions concerns/effects:
  - how knowledge is gathered
  - how the world (they know) is organized
- this results in two dimensions:
  - subjective-objective (w.r.t. knowledge)
  - conflict-order (w.r.t. the world)
- which results in 4 basic positions to requirements engineering that an analyst can take

The Positions of Analyst in RE

1. functional (objective+order): the analyst is the expert who empirically seeks the truth
2. social-relativism (subjective+order): the analyst is a `change agent’. RE is a learning process guided by the analyst
3. radical-structuralism (objective+conflict): there is a struggle between classes; the analyst chooses for either party
4. neohumanism (subjective+conflict): the analyst is kind of a social therapist, bringing parties together

Four positions to RE
Functional

- **Functional**: functional (objective+order): the analyst is the expert who empirically seeks the truth
  - Conflicts are solved by management.
  - Management tells how things go.
  - Requirements are well-articulated, shared, objective.
  - Development is formal, rational.
  - Politics is irrelevant.
  - Reality is measurable, and the same to everyone.
  - Design is a technical problem.

Four positions to RE

Social-Relativism

- **Social-relativism:** (subjective+order): the analyst is a `change agent’
  - There is not one truth, there may be different perceptions.
  - An information system is part of a continually changing social environment.
  - System goals result from an organization shaping its own reality.
  - There is no objective criterion of good or bad.
  - The goal is to arrive at a consensus.

Four positions to RE
Radical-Structuralism and Neohumanism

- **Radical-structuralism**: radical-structuralism (objective+conflict):
  - There is a struggle between classes; the analyst chooses for either party
  - There is a fundamental truth, but there also is a fundamental battle between classes.
  - You may choose for either management (machines replace people, machines guide people’s work, supervision, loss of jobs, less interesting work) or for the workers (enhance labor skills, make work more interesting and challenging).
- **Neohumanism** (subjective+conflict): rather hypothetical; emancipation, remove barriers

Requirements Elicitation

- After knowing the position we play in the RE process of a project, the next thing we do is to collect the requirements.
- The activities in the requirements elicitation process can be classified into five types.
  1. Understanding the application domain
  2. Identifying the sources of requirements
  3. Analysing the stakeholders
  4. Selecting the techniques, approaches, and tools to use
  5. Eliciting the requirements from stakeholders and other sources

  Since Elicitation is an iterative process, we cannot finish one type before starting another.

Requirements Elicitation
1. Understanding the application domain

- Start from understanding the application domain

- The current environment of the system needs to be thoroughly explored including the political, organizational, and social aspects related to the system, in addition to any constraints they may enforce upon the system and its development.

Requirements Elicitation
2. Identifying the sources of requirements

- Requirements spread across many sources and exist in different formats.
- People
  - Stakeholders: the main source of requirements for the system.
  - Users and subject matter experts: supply details and needs
- Things
  - E.g., Existing systems and operation processes
  - E.g., Existing documentation (e.g., manuals, forms, and reports):
  - provide useful information about the organization, environment, and rationales

Requirements Elicitation
3. Analyzing the stakeholders

- Stakeholders are people who have an interest in the system
  - Groups and individuals internal and external to the organization.
  - Customers and direct users of the system
- Analyze and involve all the relevant stakeholders.
  - The process of analyzing the stakeholders also often includes identifying key user representatives and product champions (who support the software).
- Consult them during requirements elicitation.

Also see: https://www.forbes.com/advisor/business/what-is-stakeholder-analysis/

Requirements Elicitation

4. Selecting the techniques, approaches, and tools to use

- Requirements elicitation is best performed using a variety of techniques (e.g., interviewing and form analysis).
  - In many projects, several elicitation techniques are employed during and at different software development life cycle stages.

Requirements Elicitation

5. Eliciting the requirements from stakeholders and other sources

- After the above four preparation steps, this step is the actual step to collect requirements.
- Targets
  - Establish the scope of the system
  - Investigate in detail the needs and wants of the stakeholders, especially the users
  - Determine the future processes the system will perform with respect to the business operations,
  - Examine how the system may support to satisfy the major objectives (and address the key problems) of the business

Requirements Elicitation

Requirements Elicitation Techniques

- **interview**
- Delphi technique
- **brainstorming session**
- **task analysis**
- **scenario analysis**
- ethnography
- **form analysis**
- analysis of natural language descriptions
- questionnaire
- synthesis from existing system
- domain analysis
- Business Process Redesign (BPR)
- Prototyping
- **mind mapping**
- **group storytelling**
- **user stories**
- **Focus group**
- **Faciliated workshop**
- …

We visit the highlighted ones in the next few slides.

Requirements Elicitation
Interview

- Commonly used in requirements elicitation.
- Ask stakeholders questions about the current application’s usage and the usage of the application to be built.
- Good at getting an overall understanding of stakeholders' needs and the current application's problem from their views.
- Inherently informal, and their effectiveness depends greatly on the quality of interaction between the participants.
- Interviews provide an efficient way to collect large amounts of data quickly

Requirements Elicitation

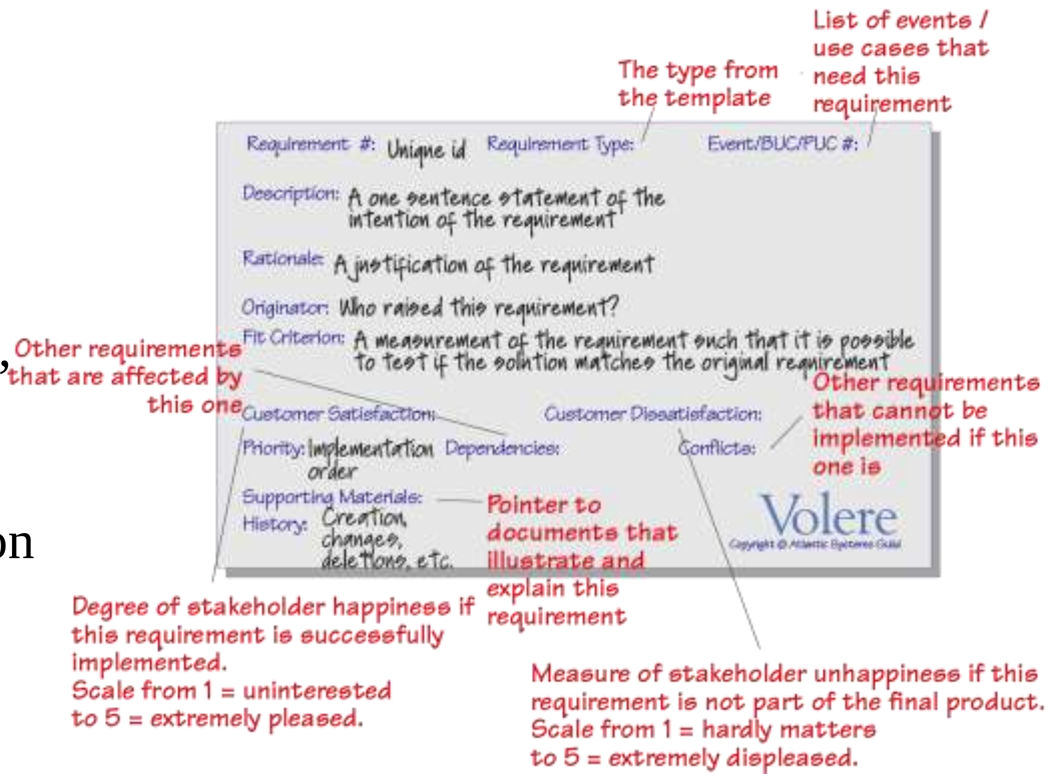
Interview

- Three types of interviews: unstructured, structured, and semi-structured (hybrid of the former two)

- Unstructured
  - Limited control over the direction of discussions.
  - Best applied for exploration when there is a limited understanding of the domain
  - Risky: Too much detail in some areas, and not enough in others

Requirements Elicitation
Interview

- Structured
  - using a predetermined set of questions to gather specific information
  - The success of structured interviews depends on knowing what are the right questions to ask, when should they be asked, and who should answer them. Templates that provide guidance on structured interviews for requirements elicitation, such as Volere cards, can be used.
  - Questions can be open-ended first then close-ended (more specific).



Requirement #: Unique id Requirement Type: Event/BUC/PUC #:

Description: A one sentence statement of the intention of the requirement

Rationale: A justification of the requirement

Originator: Who raised this requirement?

Fit Criterion: A measurement of the requirement such that it is possible to test if the solution matches the original requirement

Customer Satisfaction: Customer Dissatisfaction:

Priority: Implementation order Dependencies: Conflicts:

Supporting Materials:

History: Creation, changes, deletions, etc.

Volere

Copyright © Atlantic Systems Guild

The type from the template

List of events / use cases that need this requirement

Other requirements that are affected by this one

Other requirements that cannot be implemented if this one is

Pointer to documents that illustrate and explain this requirement

Degree of stakeholder happiness if this requirement is successfully implemented.
Scale from 1 = uninterested
to 5 = extremely pleased.

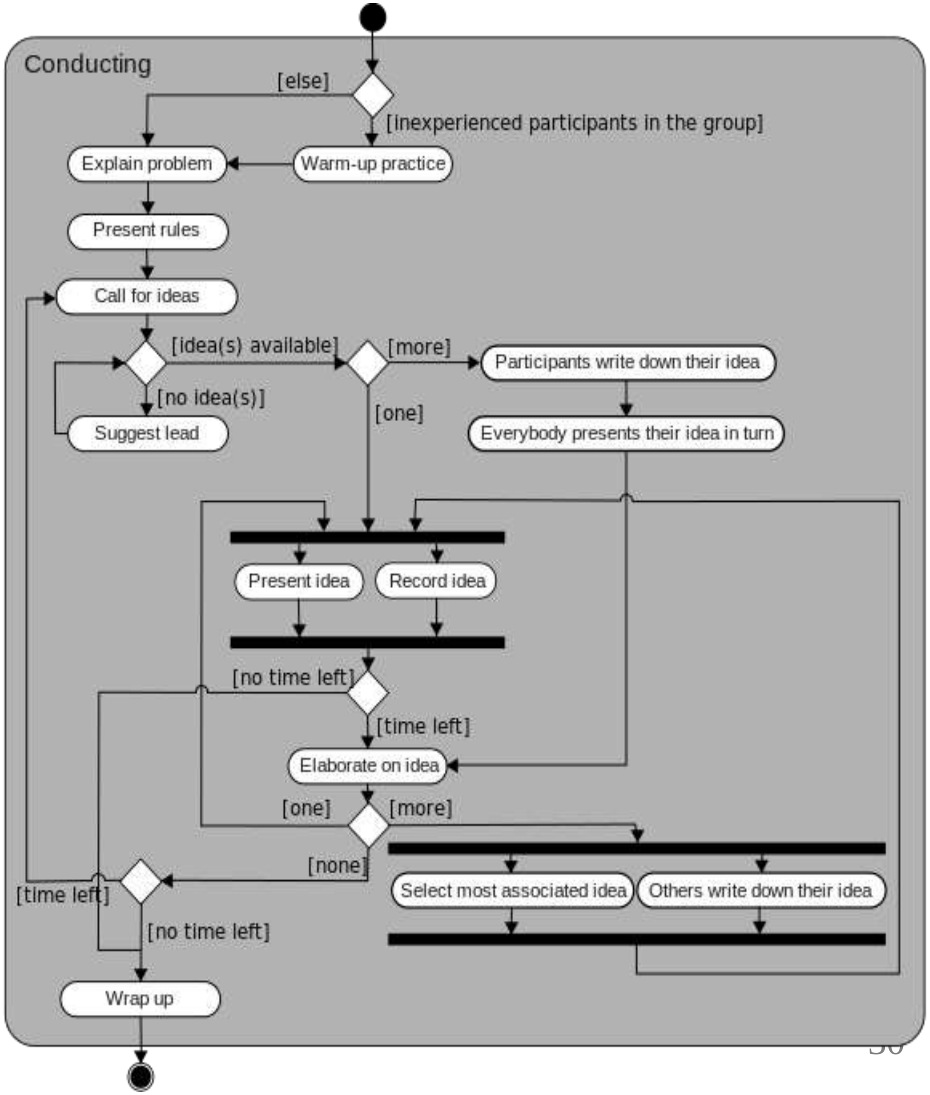
Measure of stakeholder unhappiness if this requirement is not part of the final product.
Scale from 1 = hardly matters
to 5 = extremely displeased.

Requirements Elicitation

Brainstorming session

Osborn's method

- **Brainstorming** is a group-based activity by gathering a list of ideas iteratively toward a particular topic



Requirements Elicitation
Task Analysis

- Task analysis analyzes how people perform their jobs:
  - the things they do, the things they act on, and the things they need to know.
- It employs a top-down approach where high-level tasks are decomposed into subtasks and eventually detailed sequences until all actions and events are described.
  - It determines the knowledge used or required to carry tasks out.
- E.g., how a SGS staff completes the student enrolment task, in which many typical and atypical situations need to be handled.

Requirements Elicitation

Scenario-Based Analysis

- Provides a more user-oriented perspective on designing and developing an interactive system.

- E.g., observe how the SGS staff handles the enrolments of a non-local student A and a local student B. Then, ask questions on cases not yet covered.

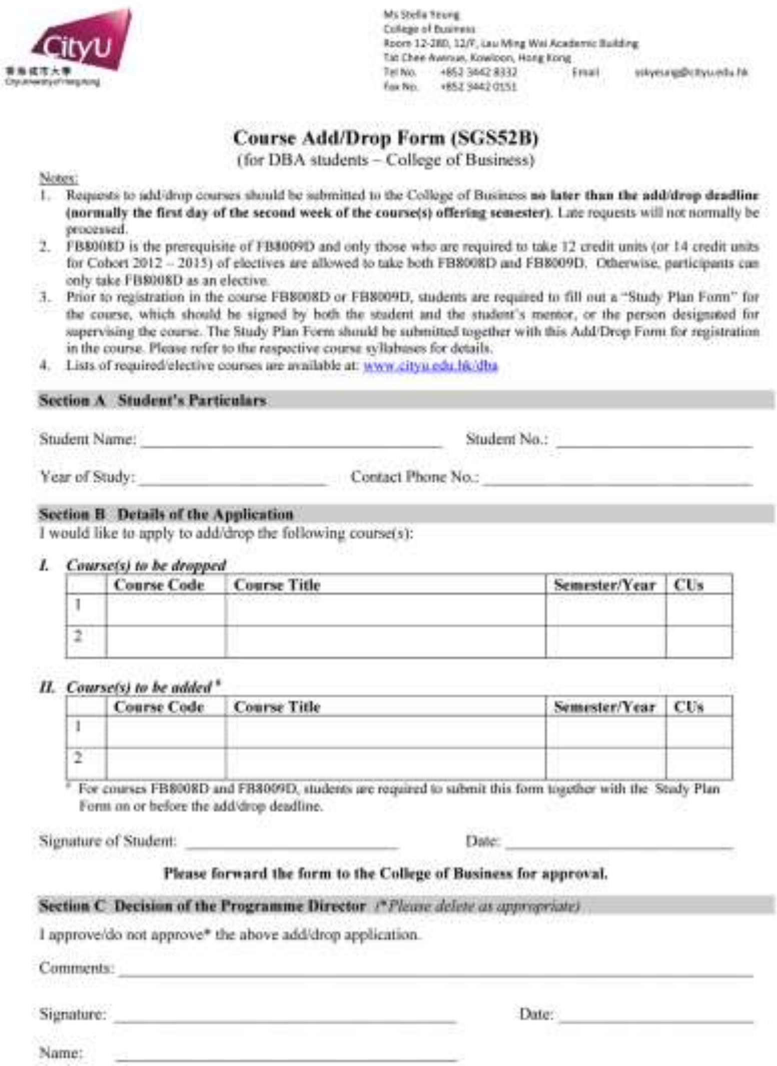
Requirements Elicitation
Scenario-Based Analysis (example)

- first shot by observation on book return in Library.
  1. check due back date
  2. if overdue, collect fine
  3. record book as being available again
  4. put book back
- followed by a discussion with the librarian
  - what if the person returning the book is not registered as a client?
  - what if the book is damaged?
  - how to handle in case the client has other books that are overdue, and/or an outstanding reservation?

Requirements Elicitation

Form analysis

- Figure out items that are certain or have variety, and the time (past, while filling the form, or future) that the information for the item is available.



CityU
香港城市大學
City University of Hong Kong

Ms Stella Yeung
College of Business
Room 12-280, 12/F, Lau Ming Wai Academic Building
Tat Chee Avenue, Kowloon, Hong Kong
Tel No. +852 3442 8332 Email sikyeung@cityu.edu.hk
Fax No. +852 3442 0151

Course Add/Drop Form (SGS52B)
(for DBA students – College of Business)

Notes:

1. Requests to add/drop courses should be submitted to the College of Business **no later than the add/drop deadline (normally the first day of the second week of the course(s) offering semester)**. Late requests will not normally be processed.
2. FB8008D is the prerequisite of FB8009D and only those who are required to take 12 credit units (or 14 credit units for Cohort 2012 – 2015) of electives are allowed to take both FB8008D and FB8009D. Otherwise, participants can only take FB8008D as an elective.
3. Prior to registration in the course FB8008D or FB8009D, students are required to fill out a "Study Plan Form" for the course, which should be signed by both the student and the student's mentor, or the person designated for supervising the course. The Study Plan Form should be submitted together with this Add/Drop Form for registration in the course. Please refer to the respective course syllabuses for details.
4. Lists of required/elective courses are available at: www.cityu.edu.hk/dba

Section A Student's Particulars

Student Name: ____________________ Student No.: ____________

Year of Study: ____________ Contact Phone No.: ____________________

Section B Details of the Application
I would like to apply to add/drop the following course(s):

I. Course(s) to be dropped

	Course Code	Course Title	Semester/Year	CUs
1				
2				

II. Course(s) to be added*

	Course Code	Course Title	Semester/Year	CUs
1				
2				

* For courses FB8008D and FB8009D, students are required to submit this form together with the Study Plan Form on or before the add/drop deadline.

Signature of Student: ____________________ Date: ____________________

Please forward the form to the College of Business for approval.

Section C Decision of the Programme Director *(*Please delete as appropriate)*

I approve/do not approve* the above add/drop application.

Comments: ____

Signature: ____________________ Date: ____________

Name: ____________________

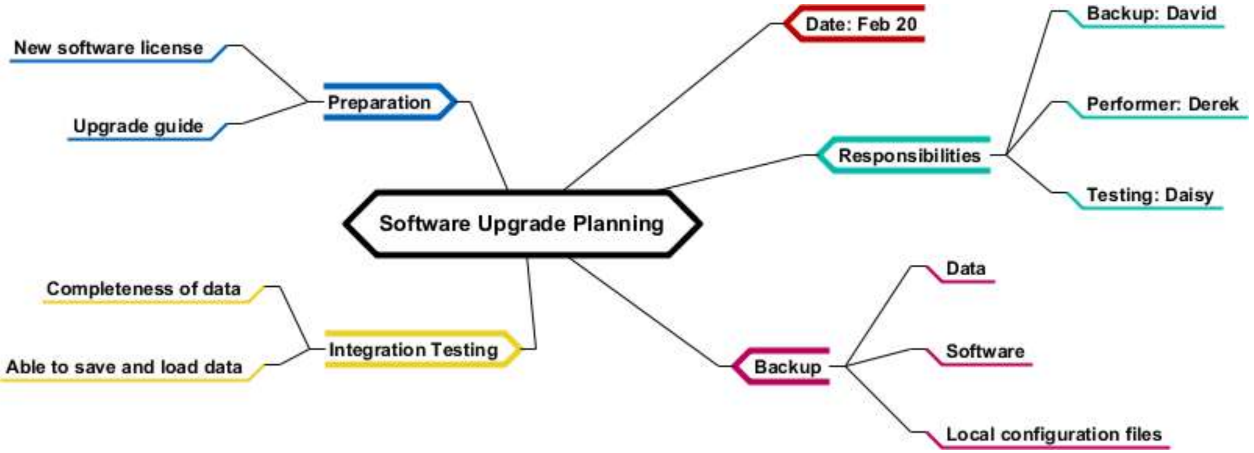
Requirements Elicitation

Focus Group and Facilitated Workshop

- A **focus group** is a small, but demographically diverse, group of people whose reactions are studied in guided or open discussions.
  - people are asked about their perceptions, opinions, beliefs, and attitudes towards
  - RE Analyst records the vital points he or she is getting from the group.
- A **facilitated workshop** involves cross-functional team members to study the topic from all perspectives and make a decision as the outcome of the workshop.

Requirements Elicitation
Mind mapping, group storytelling, user stories

- Mind map



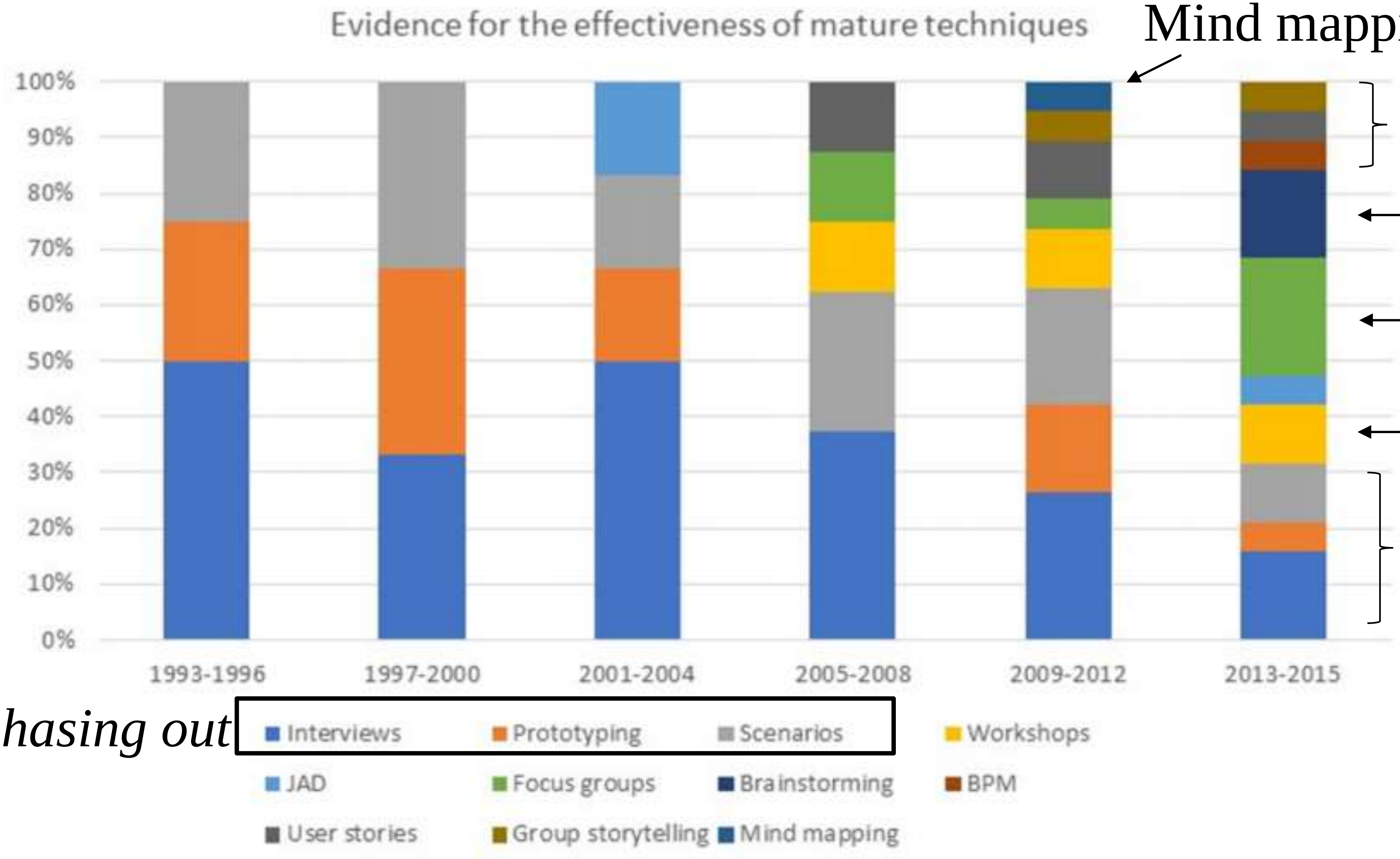
- Group storytelling
  - Co-construct a story, share a story, complete an unfinished story provided by a facilitator, zoom-in/out or summarize a story, roleplay a story, analyze a story
- User story
  - E.g., As a manager, I want to browse my existing quizzes so I can recall what I have in place and figure out if I can just reuse or update an existing quiz for the position I need now

Elicitation Technique Selection: How Do Experts Do It? [4]

- **Interviewing**: particularly useful to gather initial background information when working on new projects in new domains.
- **Collaborative Sessions (focus group, workshop, brainstorming)**: Effective
- **Data Gathering from Existing Systems**: must do, but not over-analyze
- **Agile (mind mapping, group storytelling, user stories):** popular nowadays.
- **Questionnaires**: Not effective

Requirements Elicitation

Effectiveness of mature techniques [5]



Mind mapping. Out?

emerging

Brainstorming ✓

Focus group ✓

workshop ✓

Phasing out

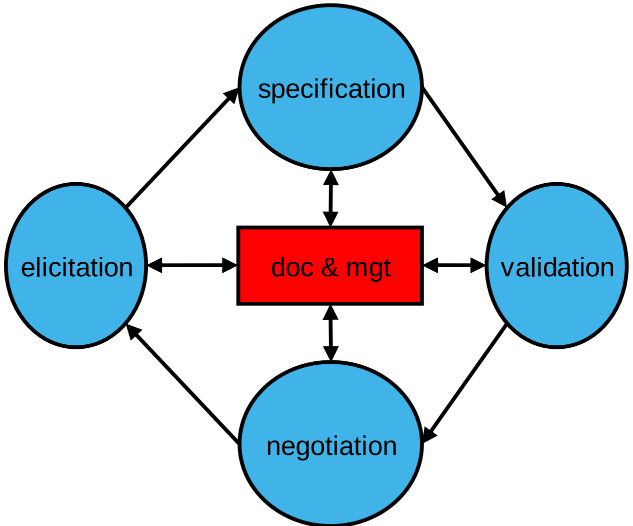
Phasing out

Recall?
Requirements engineering, main steps

Four major steps in an RE process

1. ~~understanding the problem: **elicitation**~~ Done
2. describing the problem: **specification**
3. agreeing upon the nature of the problem: **validation**
4. agreeing upon the boundaries of the problem: **negotiation**

This is an iterative process



Specifying requirements

Structuring a set of requirements

While we collect the requirements, we should structure them.

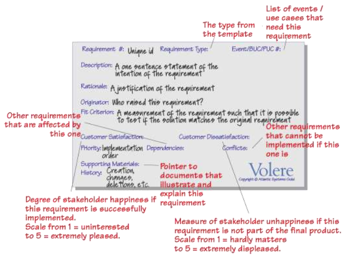
1. Hierarchical structure: Higher-level requirements are decomposed into lower-level requirements

2. Link requirements to specific stakeholders (e.g., management and end users each have their own set)

So, while we structure the requirements, we also analyze them and their relationships.

Specifying requirements

Example of Structuring Requirements



List of events / use cases that need this requirement

The type from the template

Requirement #: Unique id Requirement Type: Event/BUC/PUC #:

Description: A one sentence statement of the intention of the requirement

Rationale: A justification of the requirement

Originator: Who raised this requirement?

Fit Criterion: A measurement of the requirement such that it is possible to test if the solution matches the original requirement

Other requirements that are affected by this one

Customer Satisfaction: Customer Dissatisfaction:

Priority: Implementation order Dependencies: Conflicts:

Other requirements that cannot be implemented if this one is

Supporting Materials:

History: Creation, changes, deletions, etc.

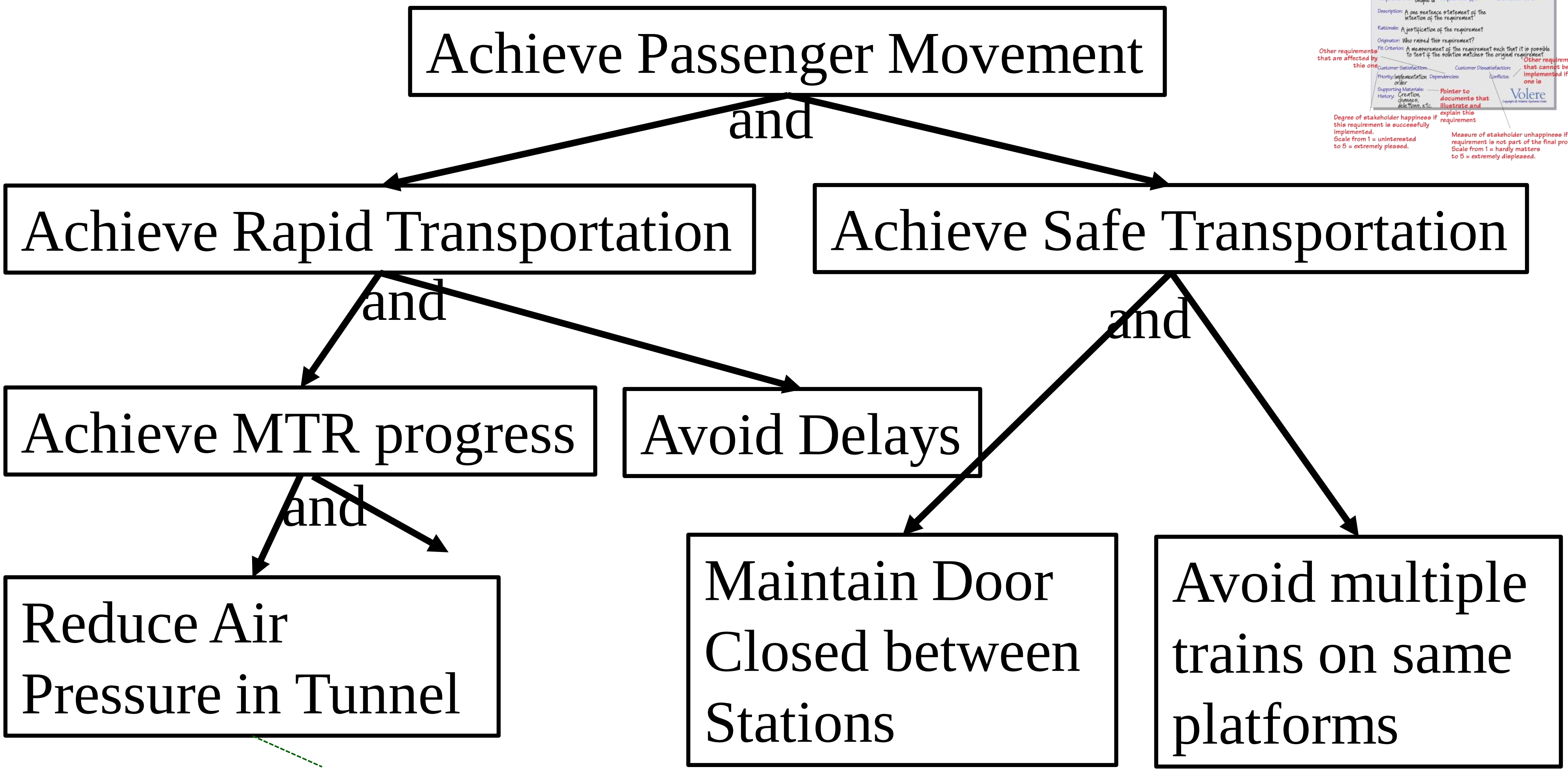
Pointer to documents that illustrate and explain this requirement

Volere

Copyright © Atlantic Systems Guild

Degree of stakeholder happiness if this requirement is successfully implemented.
Scale from 1 = uninterested
to 5 = extremely pleased.

Measure of stakeholder unhappiness if this requirement is not part of the final product.
Scale from 1 = hardly matters
to 5 = extremely displeased.



Achieve Passenger Movement

and

Achieve Rapid Transportation

Achieve Safe Transportation

and

and

Achieve MTR progress

Avoid Delays

and

Reduce Air Pressure in Tunnel

Maintain Door Closed between Stations

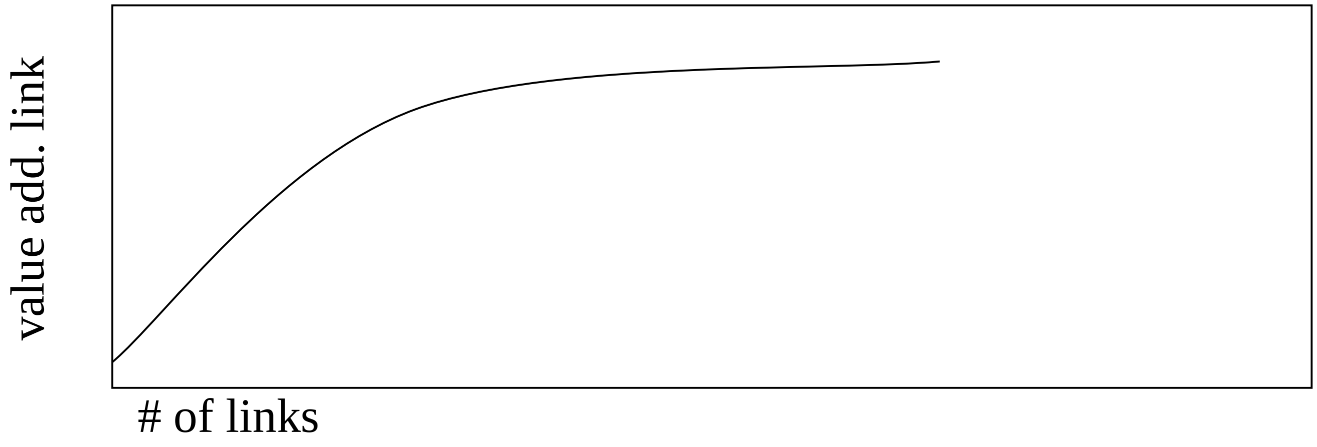
Avoid multiple trains on same platforms

Who concerns this? (can express as a user story)

Validating Requirement

Direct versus indirect links

- Same requirements from multiple sources help us reason the validity of the requirements.
  - Lesson 1: don’t rely too much on indirect links (intermediaries, surrogate users)
  - Lesson 2: The more links, the better (but up to a point)



Validating Requirement

Things to look at

- Inspection of the requirement specification w.r.t. correctness, completeness, consistency, accuracy, readability, and testability.

- some aids at different stages of development
  - Early structured walkthroughs with customers
  - Prototypes of initial versions
  - Test plan or unit testing in Agile development
  - User acceptance testing when delivery

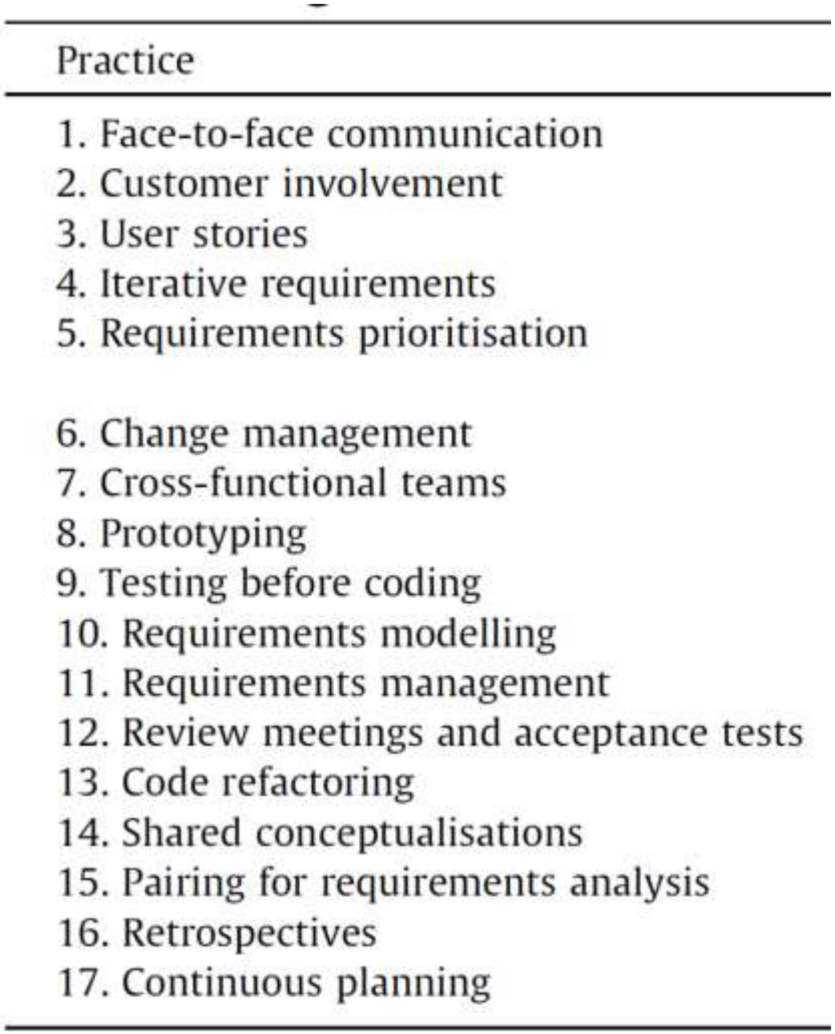
Negotiating requirements

Prioritizing Requirements

- Apart from negotiating requirements during requirement elicitation, we should also negotiate requirements with stakeholders when prioritizing them.

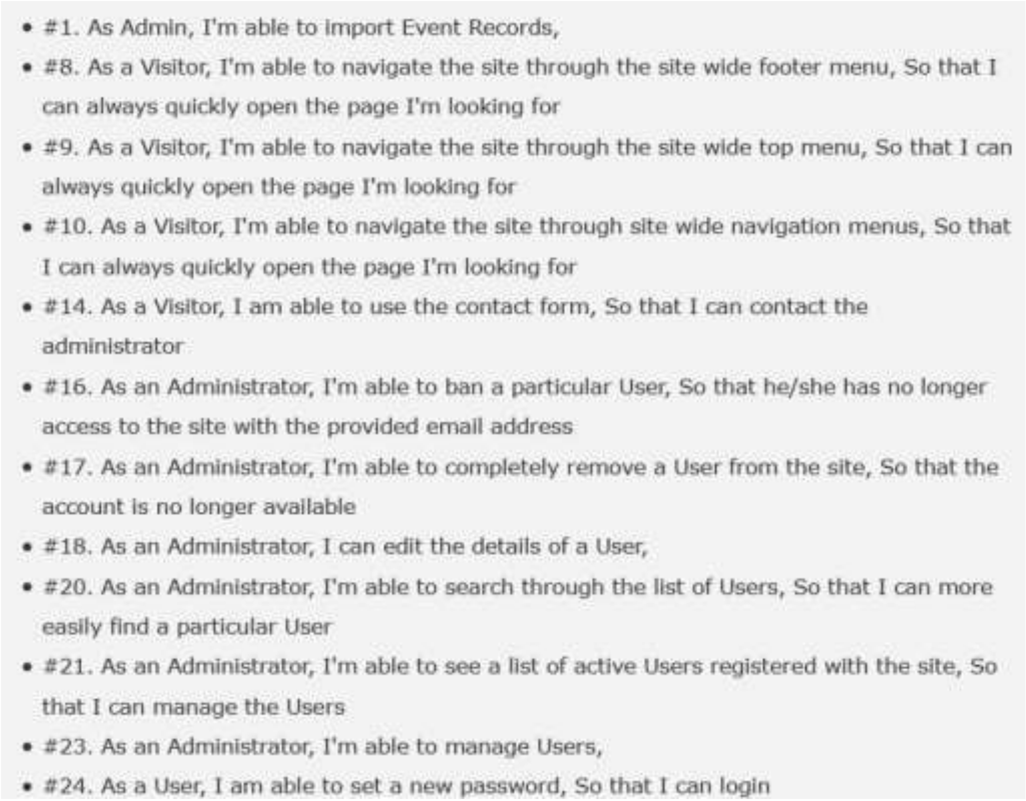
- E.g., label a requirement item as “high”, “medium”, or “low” priority,
- E.g., Classify it using the MoSCoW Method
  - **Must haves**: mandatory requirements
  - **Should haves**: important but not mandatory
  - **Could haves**: if time allows
  - **Won’t haves**: not today (may be tomorrow)

Example Practices in Agile Requirement Management



Practice
1. Face-to-face communication
2. Customer involvement
3. User stories
4. Iterative requirements
5. Requirements prioritisation
6. Change management
7. Cross-functional teams
8. Prototyping
9. Testing before coding
10. Requirements modelling
11. Requirements management
12. Review meetings and acceptance tests
13. Code refactoring
14. Shared conceptualisations
15. Pairing for requirements analysis
16. Retrospectives
17. Continuous planning

Example user stories in brief



- #1. As Admin, I'm able to import Event Records,
- #8. As a Visitor, I'm able to navigate the site through the site wide footer menu, So that I can always quickly open the page I'm looking for
- #9. As a Visitor, I'm able to navigate the site through the site wide top menu, So that I can always quickly open the page I'm looking for
- #10. As a Visitor, I'm able to navigate the site through site wide navigation menus, So that I can always quickly open the page I'm looking for
- #14. As a Visitor, I am able to use the contact form, So that I can contact the administrator
- #16. As an Administrator, I'm able to ban a particular User, So that he/she has no longer access to the site with the provided email address
- #17. As an Administrator, I'm able to completely remove a User from the site, So that the account is no longer available
- #18. As an Administrator, I can edit the details of a User,
- #20. As an Administrator, I'm able to search through the list of Users, So that I can more easily find a particular User
- #21. As an Administrator, I'm able to see a list of active Users registered with the site, So that I can manage the Users
- #23. As an Administrator, I'm able to manage Users,
- #24. As a User, I am able to set a new password, So that I can login

Summary

- RE involves elicitation, specification, validation, and negotiation
- Elicitation has five iterative steps, four of which are preparation steps. We use multiple elicitation techniques.
- We not only collect the requirements but also shape them
  - Analyzing requirements by structuring them
  - Negotiating with stakeholders in requirement elicitation and prioritization
  - Validating requirements from multiple sources

References and Resources

1. Sommerville, Software Engineering, 10th Edition. (Chapter 4)
2. Irum Inayat, Siti Salwah Salim, Sabrina Marczak, Maya Daneva, and Shahaboddin Shamshirband. 2015. A systematic literature review on agile requirements engineering practices and challenges. Comput. Hum. Behav. 51, PB (October 2015), 915-929. DOI: http://dx.doi.org/10.1016/j.chb.2014.10.046 or https://www.sciencedirect.com/science/article/pii/S074756321400569X
3. Catarina Gralha, Daniela Damian, Anthony I. (Tony) Wasserman, Miguel Goulão, and João Araújo. 2018. The evolution of requirements practices in software startups. In Proceedings of the 40th International Conference on Software Engineering (ICSE '18). ACM, New York, NY, USA, 823-833. DOI: https://doi.org/10.1145/3180155.3180158
4. Ann M. Hickey and Alan M. Davis. 2003. Elicitation Technique Selection: How Do Experts Do It?. In Proceedings of the 11th IEEE International Conference on Requirements Engineering (RE '03). IEEE Computer Society, Washington, DC, USA, 169- http://www.cse.msu.edu/~chengb/RE-491/Papers/elicitation-hickey-davis.pdf
5. Carla L. Pacheco, Ivan A. Garcia, Miryam Reyes: Requirements elicitation techniques: a systematic literature review based on the maturity of the techniques. IET Software 12(4): 365-378 (2018)
6. Martin Glinz: On Non-Functional Requirements. RE 2007: 21-26, 2007.
7. Zijad Kurtanovic, Walid Maalej: Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning. RE 2017: 490-495
8. Dewi Mairiza, Didar Zowghi, Nur Nurmuliani: An investigation into the notion of non-functional requirements. SAC 2010: 311-317, 2010.