U O W

Software Requirements, Specifications and Formal Methods

Dr. Shixun Huang

UNIVERSITY OF WOLLONGONG AUSTRALIA

Requirements Elicitation

Requirements Elicitation

- Requirements are not like fallen fruit to be simply retrieved and placed in a bushel.

The techniques include:

- Brainstorming, Card sorting, Designer as apprentice, Domain analysis, Ethnographic observation, Goal-based approaches, Group work, Interviews, Joint application development (JAD), Laddering, Prototyping, Quality function deployment (QFD), Questionnaires, Repertory grids, Scenarios, Task analysis, Use cases, User stories, Viewpoints, Workshops and etc.

Prepare for the Elicitation

- Identify all customers and stakeholders.
- Partition customers and other stakeholders groups into classes according to interests, scope, authorization, or other discriminating factors (some classes may need *multiple levels* of partitioning).
- Select a champion or representative group for each user class and stakeholder group.
- Select the appropriate technique(s) to solicit initial inputs from each class or stakeholder group.

Brainstorming

Brainstorming

- Brainstorming consists of informal sessions with stakeholders to generate overarching goals for the systems
- These kinds of meetings probably should be informal, even spontaneous, with the only structure embodying some recording of any major discoveries.
- During brainstorming sessions, some preliminary requirements may be generated
- Brainstorming is also useful for general objective setting, such as mission or vision statement generation

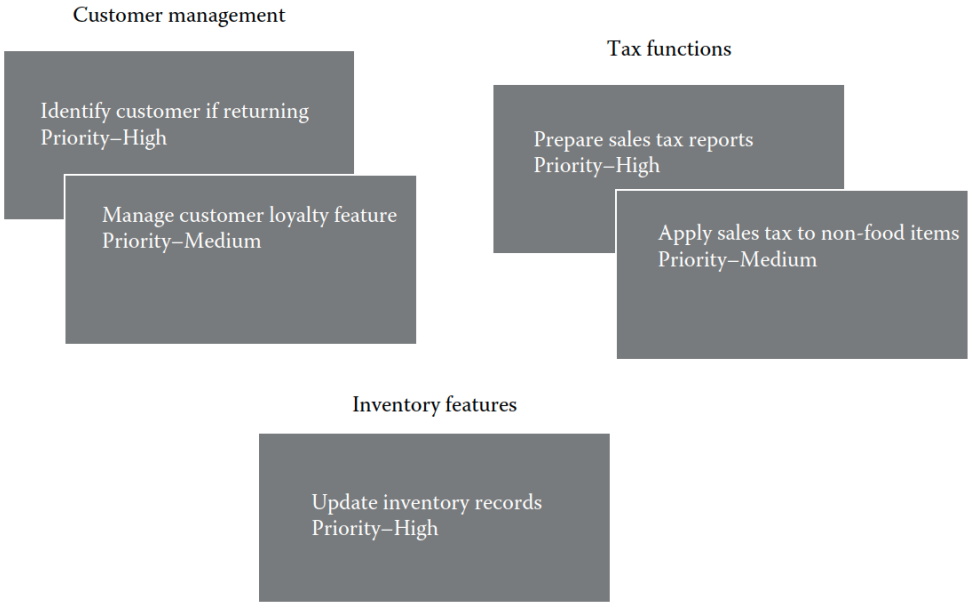
Card Sorting

Card Sorting

- Stakeholders complete a set of cards that includes key information about functionality for the system/software product
- The requirements engineer organizes these cards in some manner, generally clustering the functionalities logically.
- The sorted cards can also be used as an input to the process to develop *CRC (class, responsibility, collaboration)* cards to determine program classes in the eventual code

Card Sorting

Unsorted cards for the pet store POS system



Sorted cards for the pet store POS system

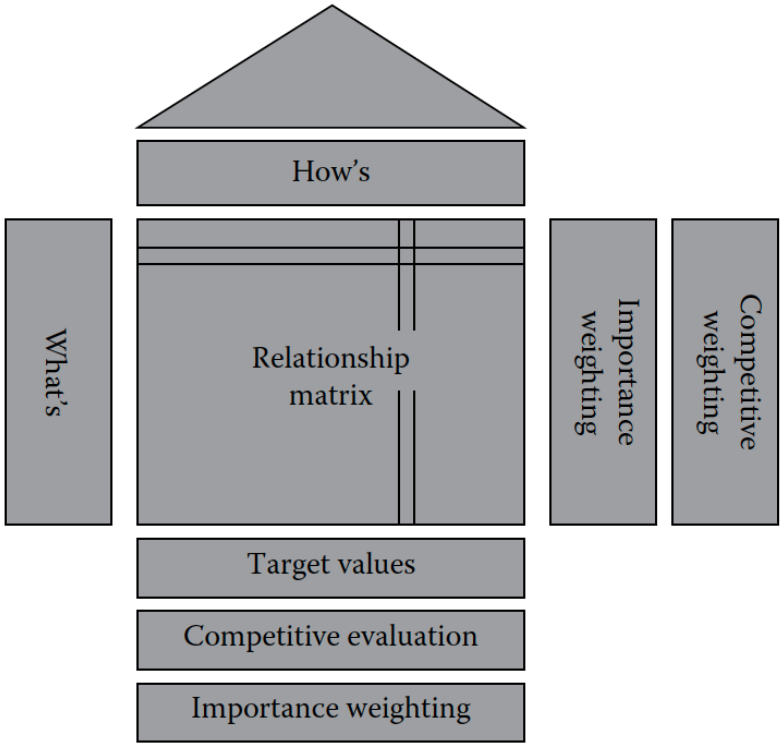
Designer as Apprentice

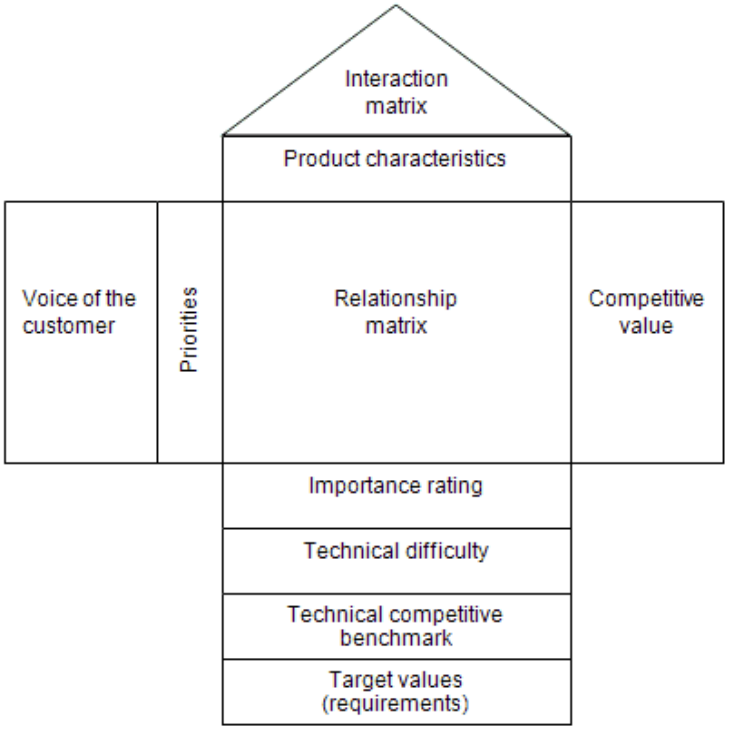
- Requirements engineer "looks over the shoulder" of the customer in order to learn enough about the customer's work to understand their needs
- Customers explain what they're doing which can help to reveal what matters in their work
- The requirements engineer must understand the structure and implication of the work, including:
  - The strategy to get work done
  - Constraints that get in the way
  - The structure of the physical environment as it supports work
  - The way work is divided
  - Recurring patterns of activity
  - The implications these have on any potential system
- Both customer and designer learn during this process

Domain Analysis

- Domain analysis involves any general approach to assess the "landscape" of related and competing applications to the system being designed
- The *Quality Function Deployment (QFD) approach* explicitly incorporates domain analysis
- QFD provides a structure for ensuring that customers' needs and desires
- The basic idea of QFD is to construct relationship matrices between customer needs, technical requirements, and priorities
- Because these relationship matrices are often represented as the roof, ceiling, and sides of a house, QFD is sometimes referred to as the "*house of quality*".

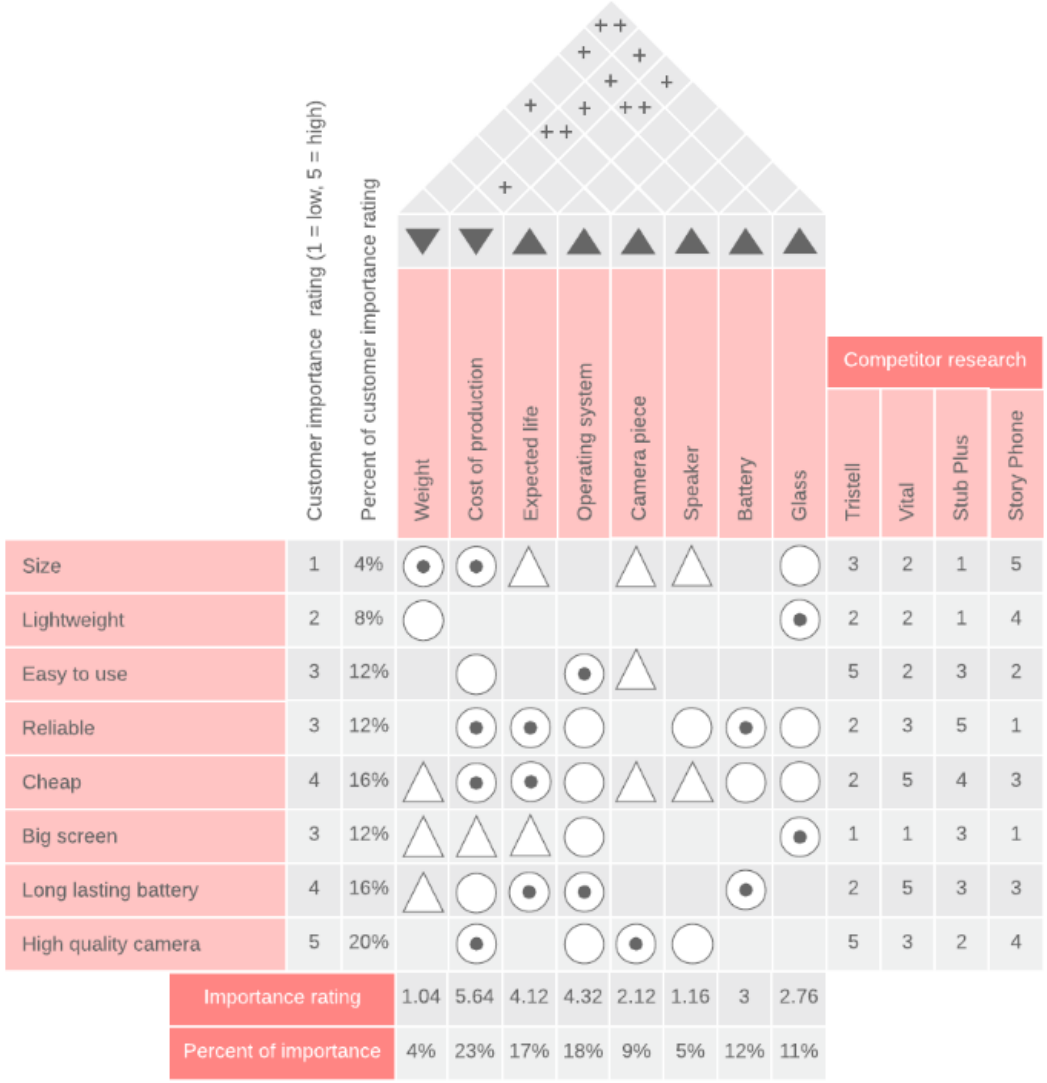
Domain Analysis





House of Quality

House of Quality Example



	Customer importance rating (1 = low, 5 = high)	Percent of customer importance rating	Weight	Cost of production	Expected life	Operating system	Camera piece	Speaker	Battery	Glass	Competitor research: Tristell	Vital	Stub Plus	Story Phone
Size	1	4%	●	●	△		△	△		○	3	2	1	5
Lightweight	2	8%	○							●	2	2	1	4
Easy to use	3	12%		○		●	△				5	2	3	2
Reliable	3	12%		●	●	○		○	●	○	2	3	5	1
Cheap	4	16%	△	●	●	○	△	△	○	○	2	5	4	3
Big screen	3	12%	△	△	△	○				●	1	1	3	1
Long lasting battery	4	16%	△	○	●	●			●		2	5	3	3
High quality camera	5	20%		●		○	●	○			5	3	2	4
Importance rating			1.04	5.64	4.12	4.32	2.12	1.16	3	2.76				
Percent of importance			4%	23%	17%	18%	9%	5%	12%	11%				

Relationship matrix		
●	Strong	9
○	Medium	3
△	Weak	1
	No assignment	0

Correlation matrix	
+ +	Strong positive
+	Positive
–	Negative
– –	Strong negative
	Not correlated

Source: https://www.lucidchart.com/blog/qfd-house-of-quality

Domain Analysis

- QFD ensure that products are not developed from only the ‘voice of of the engineer’, but also the ‘voice of customer’.
- The following requirements engineering process is prescribed by QFD:
  - Identify stakeholder’s attributes or requirements.
  - Identify technical features of the requirements.
  - Relate the requirements to the technical features.
  - Conduct an evaluation of competing products.
  - Evaluate technical features and specify a target value, for each feature.
  - Prioritize technical features for development effort.

Domain Analysis

- Advantages of QFD
  - improves the involvement of users and managers
  - shortens the development lifecycle and improves overall project development
  - supports team involvement by structuring communication processes
  - provides a preventive tool that avoids the loss of information
- Drawbacks of QFD
  - Complex and resource-intensive
  - Limited flexibility
  - Subjectivity in customer input

Ethnographic Observation

- Ethnographic observation refers to any technique in which observation of **indirect and direct** factors inform the work of the requirements engineer
- Ethnographic observation is a technique borrowed from social science in which observations of human activity and the environment are collected
- Engineers are also in a position to collect evidence of customer needs derived from the surroundings that may not be communicated directly
- Ethnographic observation can be very time-consuming and requires substantial training of the observer

Ethnographic Observation

To illustrate this technique in practice, consider this situation in which ethnographic observation occurs:

- You are gathering requirements for a smart home for a customer.
- You spend long periods of time passively observing the customer "in action" in the current home to get nonverbal clues about wants and desires.
- You gain other information from the home itself—the books on the bookshelf, paintings on the wall, furniture styles, evidence of hobbies, signs of wear and tear on various appliances, etc.

Goal-based Approaches

- Goal-based approaches comprise any elicitation techniques in which requirements are recognized to emanate from the mission statement, i.e., from a set of goals lead to requirements
- These goals may be subdivided one or more times to obtain lower-level goals.
- Then, the lower-level goals are branched out into specific high-level requirements using a structured approach such goal-question-metric (GQM).
- Finally, the high-level requirements are used to generate lower-level ones

Goal-based Approaches

For example, consider the baggage handling system mission statement:

- To automate all aspects of baggage handling from passenger origin to destination.

The following goals might be considered to fulfill this mission:

- Goal 1: To completely automate the tracking of baggage from check-in to pick-up.
- Goal 2: To completely automate the routing of baggage from check-in counter to plane.
- Goal 3: To reduce the amount of lost luggage to 1%.
- Goal 4: The system shall be user friendly.

Goal-based Approaches

Goal:

- The system shall be user friendly

Questions:

- How easy is the system to learn to use?
- How much help does a new user need?
- How many errors does a user get?

Metrics for questions:

- The time it takes a user to learn how to perform certain functions
- The number of times a user has to use the help feature over some period of time
- The number of times a user sees an error message during certain operations over a period of time.

Group Work

- Group work is a general term for any kind of group meetings that are used during the requirements discovery, analysis, and follow-up processes.
- The most important things to remember about group works:
  - research all aspects of the organization, problems, politics, environment, and so on.
  - Publish an agenda several days and stay on it
  - Allow all to have their voices heard.
  - Look for consensus at the earliest opportunity.
  - Do not leave until all items on the agenda have received sufficient discussion.
  - Publish the minutes of the meeting within a couple of days of meeting close and allow attendees to suggest changes.

The most celebrated of group-oriented work for requirements elicitation is *joint application design (JAD)*

Joint Application Design

- JAD involves highly structured group meetings with stakeholders, and analysts focused on a specific set of problems.
- Planning for a JAD review or audit session involves three steps:
  - 1. Selecting participants
  - 2. Preparing the agenda
  - 3. Selecting a location
- Reviews and audits may include some or all of the following participants:
  - Sponsors (e.g., senior management)
  - A team leader (facilitator, independent)
  - Users and managers who have ownership of requirements and business rules
  - Scribes (i.e., meeting minutes and note takers)
  - Engineering staff

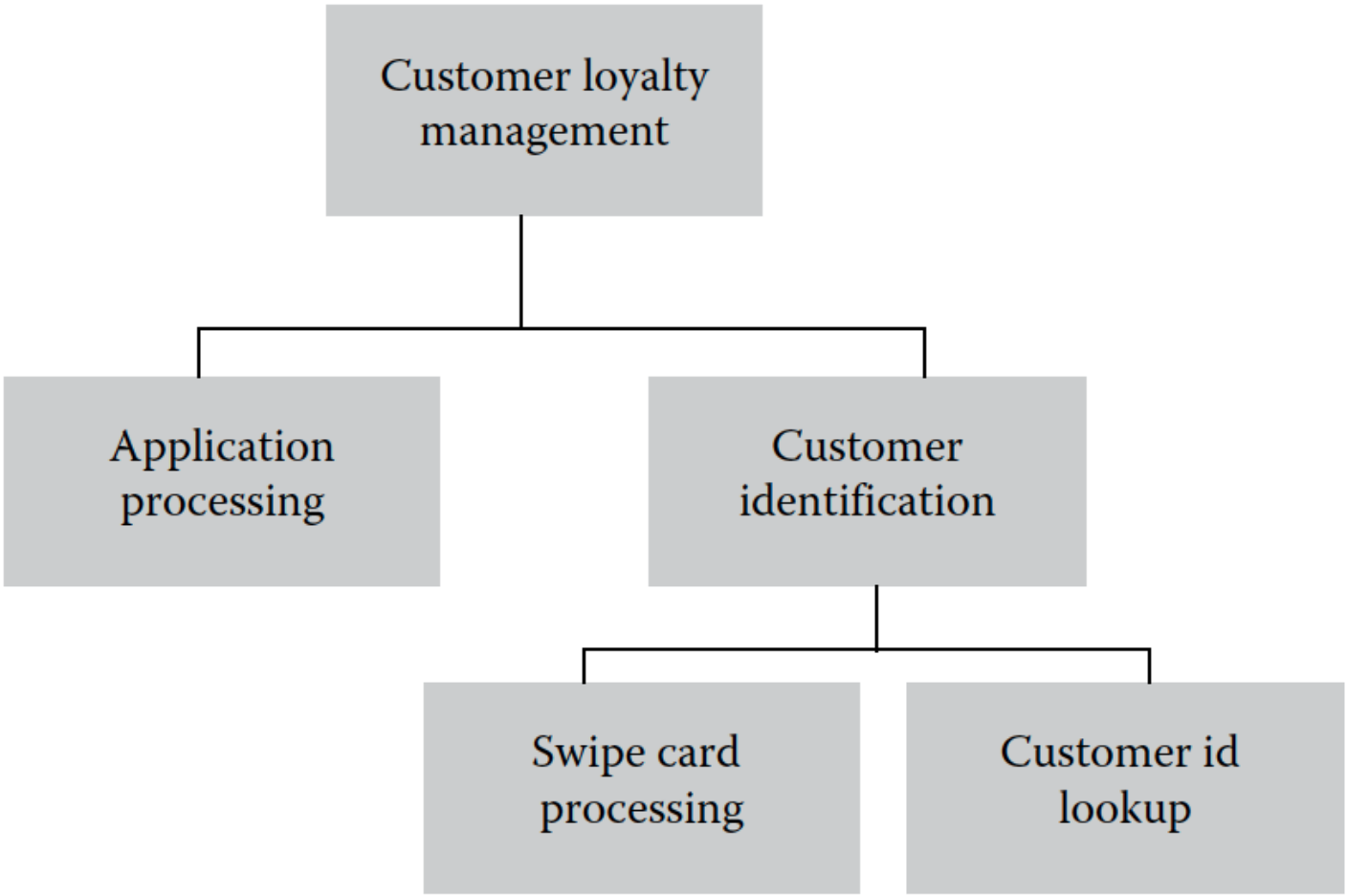
Joint Application Design

- The analyst and sponsor must determine the scope of the project and set the high-level requirements and expectations of each session.
- The session leader must also ensure that the sponsor is willing to commit people, time, and other resources to the effort.
- The agenda depends greatly on the type of review to be conducted and should be constructed to allow for sufficient time.
- The agenda, code, and documentation must also be sent to all participants well in advance of the meeting so that they have sufficient time to review them, make comments, and prepare to ask questions.
- The end product of any review session is typically a formal written document providing a summary of the items (specifications, design changes, code changes, and action items) agreed upon during the session
- The main artifact could be a first draft of the software requirement specification (SRS).

Laddering

- In laddering, the requirements engineer asks the customer short prompting questions (probes) to elicit requirements.
- Follow-up questions are then posed to dig deeper below the surface.
- The resultant information from the responses is then organized into a tree-like structure.
- For example:
  - RE: Name a key feature of the system.
  - Customer: Customer identification.
  - RE: How do you identify a customer?
  - Customer: They can swipe their loyalty card.
  - RE: What if a customer forgets their card?
  - Customer: They can be looked up by phone number.
  - RE: When do you get the customer's phone number?
  - Customer: When customers complete the application for the loyalty card.
  - RE: How do customers complete the applications? …

Laddering



Laddering diagram for the pet store POS system

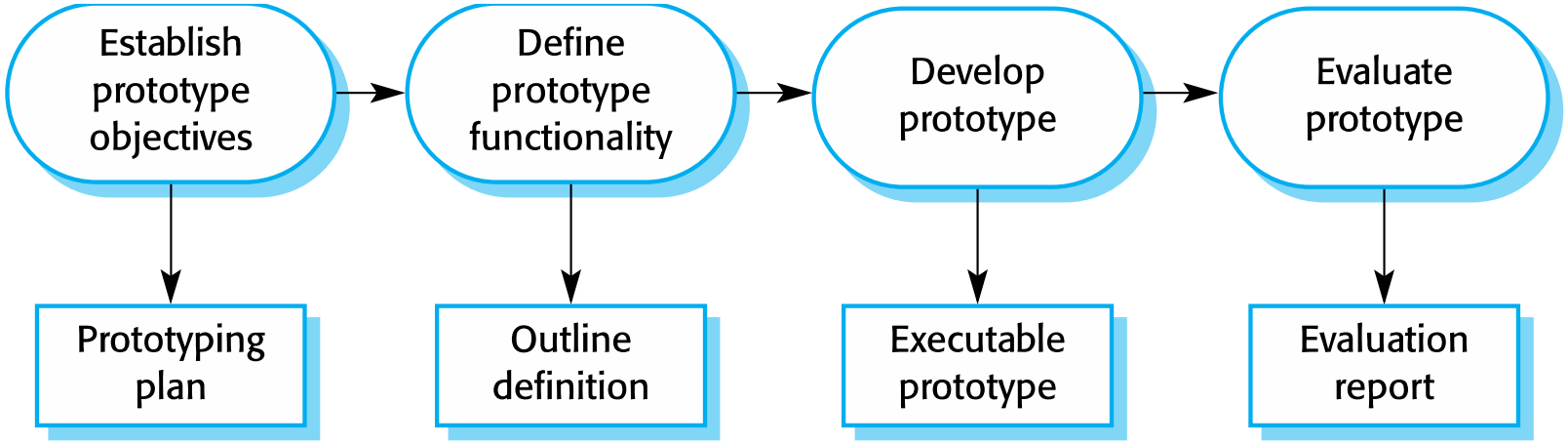
Prototyping

- Prototyping involves construction of models of the system in order to discover new features, particularly usability requirements.
- Prototyping is a particularly important technique for requirements elicitation.
- It is used extensively, for example, in the spiral software development model, and agile methodologies consist essentially of a series of increasingly functional non-throwaway prototypes.
- Prototypes can involve working models and non-working models. Working models can include executable code in the case of software systems and simulations, or temporary or to-scale prototypes for non-software systems.
- Non-working models can include storyboards and mock-ups of user interfaces.

Benefits of prototyping

- Improved system usability.
- A closer match to users' real needs.
- Improved design quality.
- Improved maintainability.
- Reduced development effort.

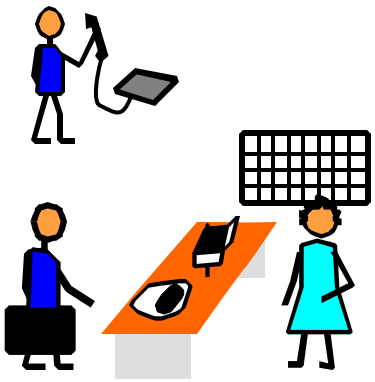
The process of prototype development

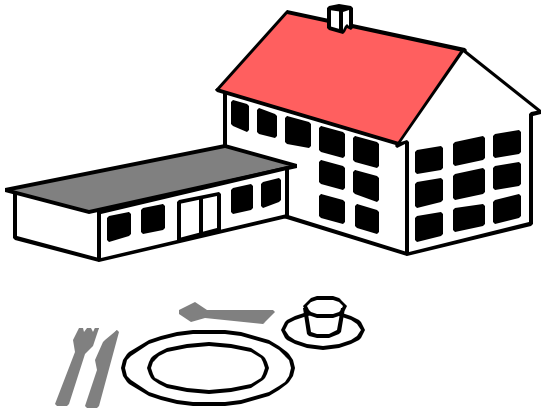


Prototype development

- May be based on rapid prototyping languages or tools
- May involve leaving out functionality
  - Prototype should focus on areas of the product that are not well-understood;
  - Error checking and recovery may not be included in the prototype;
  - Focus on functional rather than non-functional requirements such as reliability and security

Case Study of a Prototype: Hotel system





Task list

Book guest
Checkin
Checkout
Change room
Record services
Breakfast list

Breakfasts 23/9

Room	Buffet	In room
11		1
12	2	
13	1	1
15		
. . .		

User tasks

When a guest phones to book a room. The receptionist must record the name, address, for the guest, the dates he is going to stay at the hotel and the room he will stay in. We call this is the *booking task*.

When a guest arrives at the hotel, the receptionist must allocate a free room to him, give him the key and record that he now stays in this room and has to pay for it. The guest may or may not have booked the room in advance. This is the *check-in task*.

It may happen that the guest wants to change to another room. The system must also support this *change-room task*.

The first Hotel System Prototype

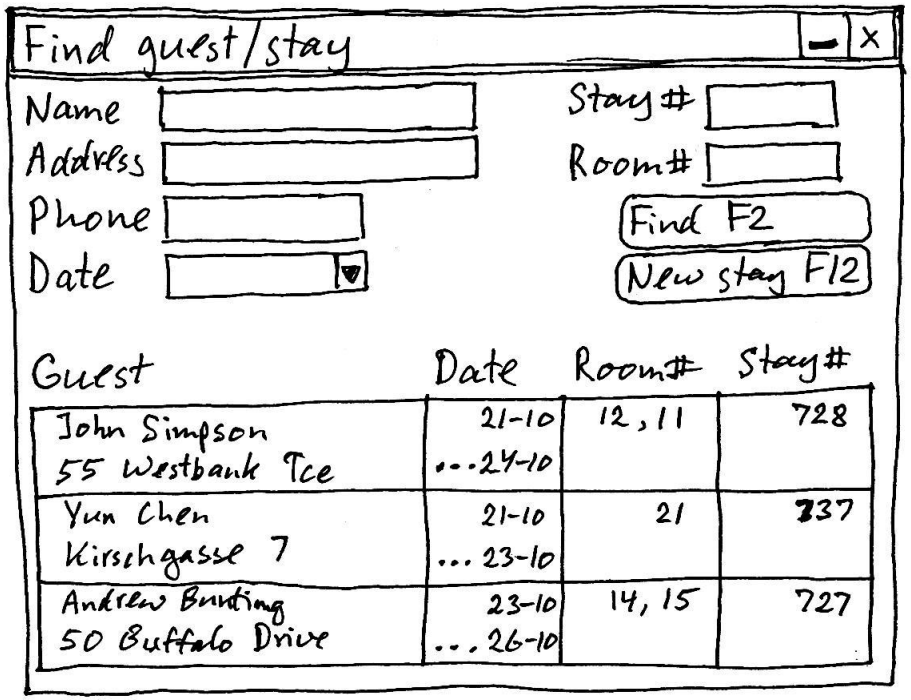
We will look at the first prototype of the hotel system and how it was suability tested and what usability problems we found.

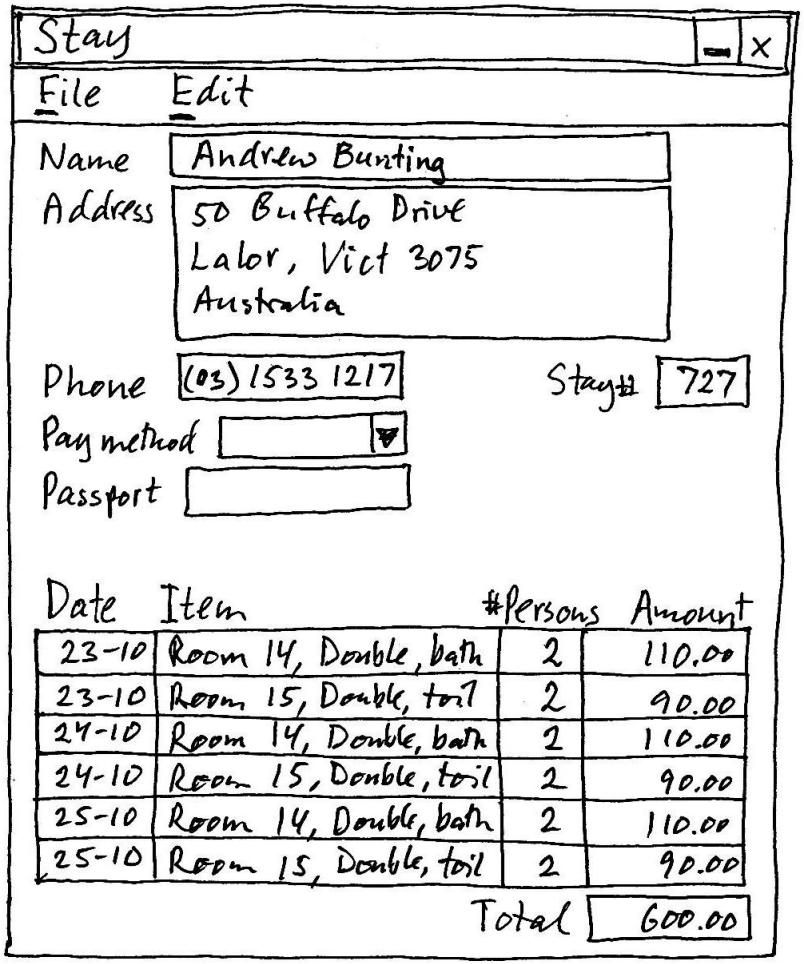
In this stage, we can prototype four screens of prototype and three menus.

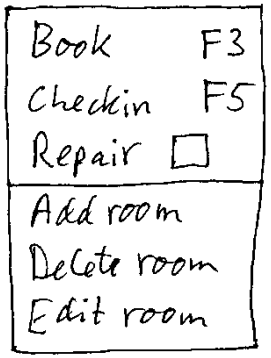
Screens

- **Find Guest Screen:** the receptionist will use the first screen to find guests, i.e. enter search criteria such as part of guest's name or address System will show a list of guests that match the searching criteria.
- **Stay and New Stay Screens:** The New stay and Stay screens are used to record a stay for a guest, for instance when he books.
- **Rooms Screen:** Use to find free rooms

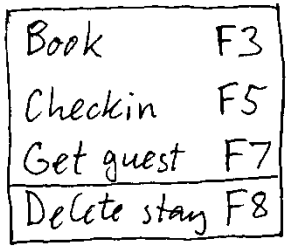
Case study: Hotel system prototype



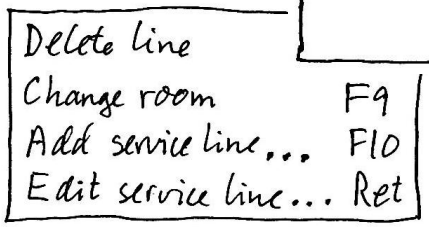




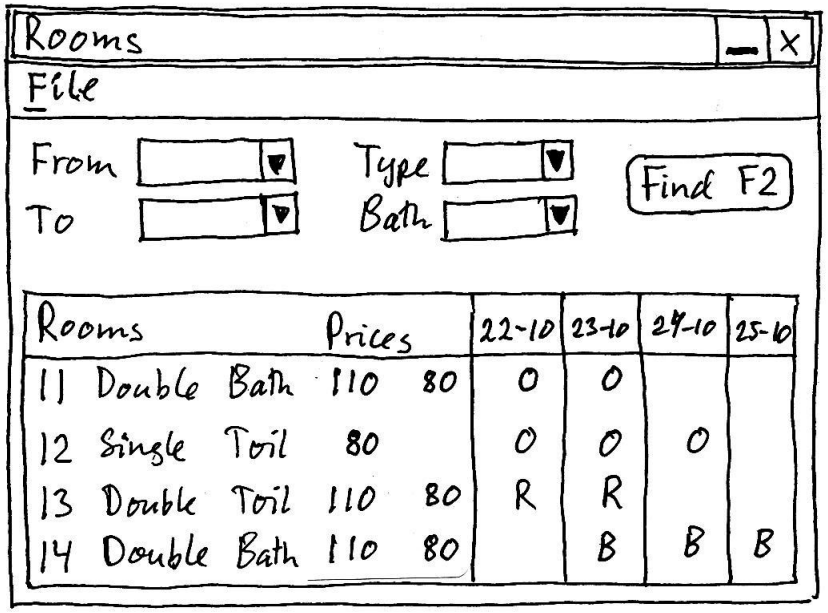
(Rooms)

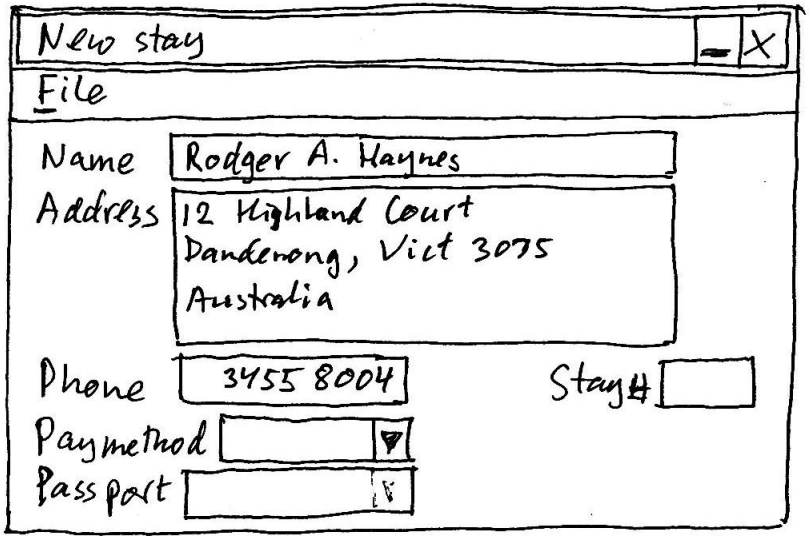


(New stay)



(Stay, Edit)





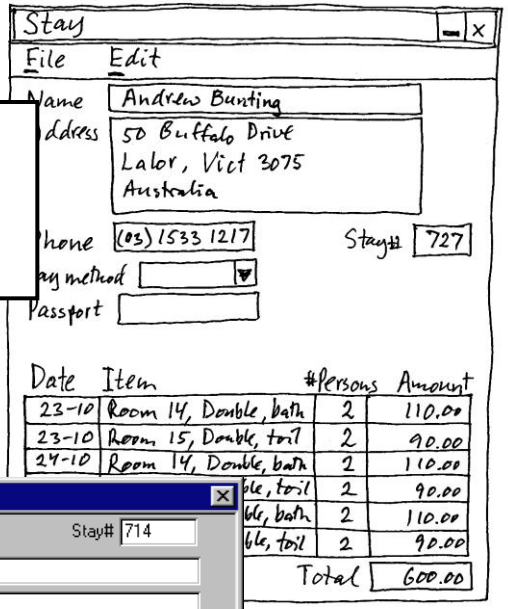
Types of Prototypes

- **Hand-draw mock-up.** The designer draws the screens by hand using paper and pencil.
- **Tool-draw mock-up.** The designer draws the screens on the computer using the same tool that will be used in the final product.
- **Screen prototype.** The screens are shown on the real computer screen, but they have little functionality.
- **Functional prototype.** It is similar to a screen prototype, but with more buttons, menus points, etc. Actually do something.

Various prototypes

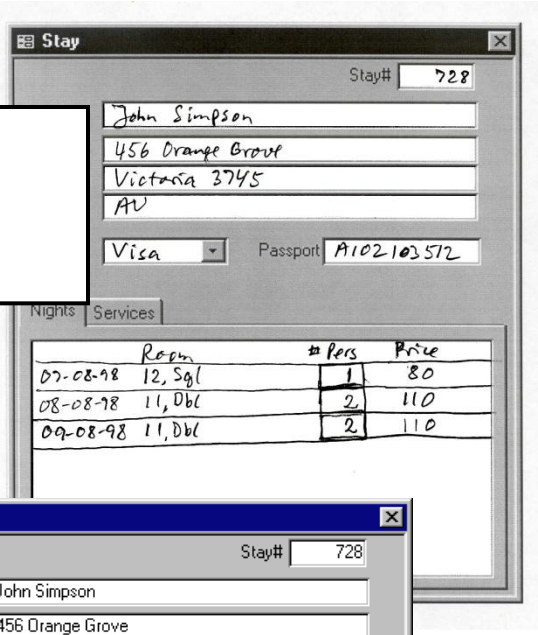
Hand-drawn mockup:

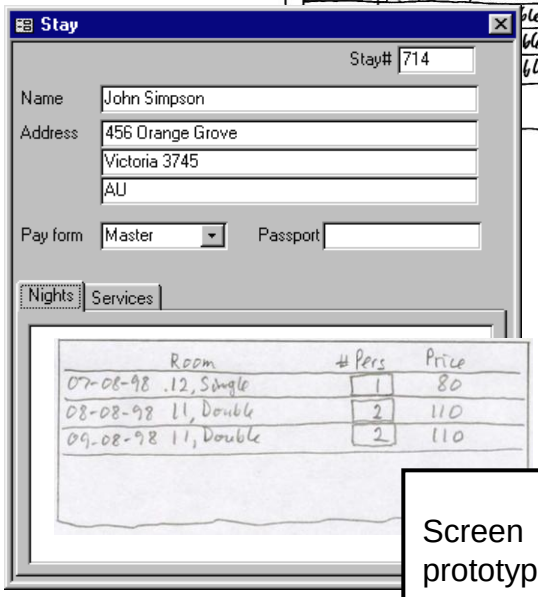
15-30 min



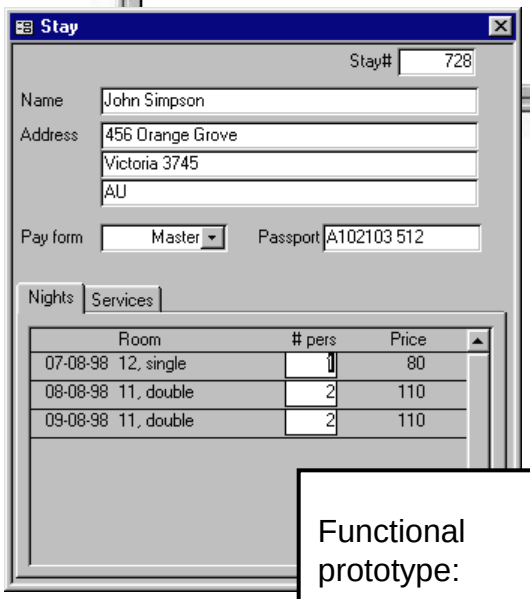
Tool-drawn mockup:

30-60 min





Which prototype is the best?



Screen prototype:

1-4 hours

Functional prototype:

2-8 hours

What is the best prototype?

The purpose of prototype in requirement engineering are to help developers to **elicit** and **validate** the system requirements.

Surprisingly, all four kinds of prototypes can detest usability problems with much same hit-rate. They are equally good for defining what to program, and for discussing with users and customers. The main difference between the prototypes is the time they take to make.

Questionnaires/Surveys

- Requirements engineers often use questionnaires and other survey instruments to reach large groups of stakeholders.
- Surveys are generally used at early stages of the elicitation process to quickly define the scope boundaries.
- Questions can be closed (e.g., multiple choice, true-false) or open-ended—involving free-form responses.
- Survey elicitation techniques are most useful *when the domain is very well understood by both stakeholders and requirements engineer*.
- For example, some possible survey questions for the pet store POS system are:
  - How many unique products (SKUs) do you carry in your inventory? (a) 0–1000; (b) 1001–10,000; (c) 10,001–100,000;
  - How many different store locations do you have? _____
  - How many unique customers do you currently have? _____

Repertory Grids

- Repertory grids incorporate a structured ranking system for various features of the different entities in the system, and are typically used when the customers are domain experts.
- Repertory grids are particularly useful for identification of agreement and disagreement within stakeholder groups.
- The grids look like a feature or quality matrix in which rows represent system entities and desirable qualities, and columns represent rankings based on each of the stakeholders.
- In essence, these ratings reflect the agendas or differing viewpoints of the stakeholders.

Baggage handling speed	1	1	5
Fault-tolerance	4	5	5
Safety	5	4	4
Reliability	3	5	5
Ease of maintenance	3	5	5
	Airline worker's union rep	Maintenance engineer	Airport operations manager

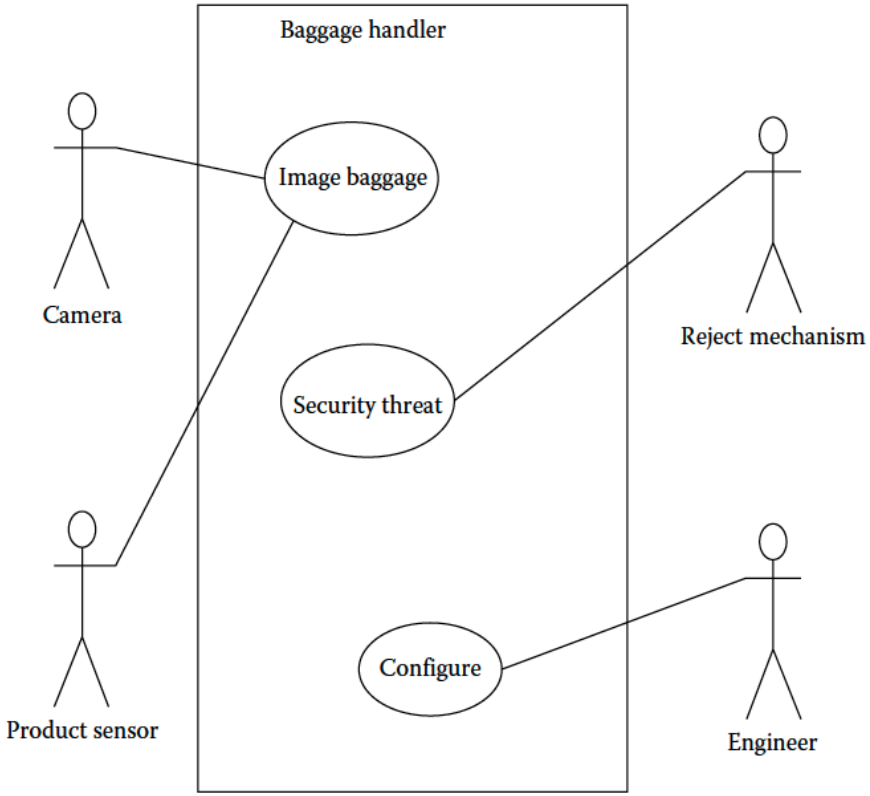
Partial repertory grid for the baggage handling system

User Story

- User stories are short conversational texts that are used for initial requirements discovery and project planning.
- User stories are widely employed in conjunction with agile methodologies.
- User stories are written by the customers in terms of what the system needs to do for them and in their own "voice." – Who can do What.
- An example of a user story for the pet store POS system is as follows:
  - Each customer should be able to easily check out at a register.
  - Self-service shall be supported.
  - All coupons, discounts, and refunds should be handled this way.

Use Case

- Use cases depict the interactions between the system and the environment around the system, human users and other systems.
- Use cases describe scenarios of operation of the system from the designer's perspective



Use case diagram of baggage inspection system

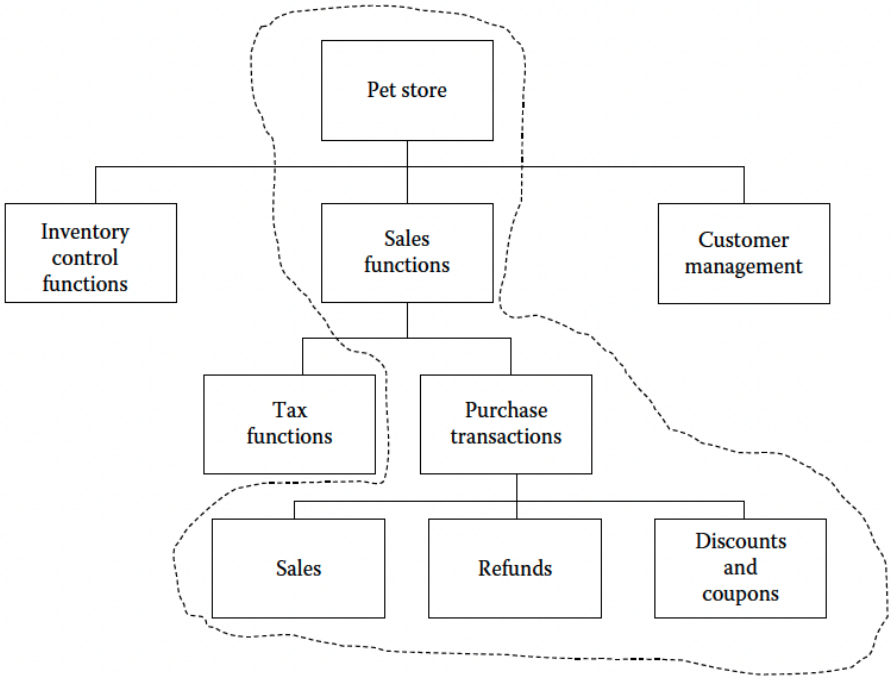
Scenarios

- Scenarios are informal descriptions of the system in use that provide a high-level description of system operation, classes of users, and exceptional situations.
- Scenarios are quite useful when the domain is novel
- Here is a sample scenario for the pet store POS system.

A customer walks into the pet store and fills the cart with a variety of items. When checking out, the cashier asks if the customer has a loyalty card. If so, the cashier swipes the card, authenticating the customer. If not, then the cashier offers to complete one on the spot. After the loyalty card activity, the cashier scans products using a bar code reader. As each item is scanned, the sale is totaled and the inventory is appropriately updated. Upon completion of product scanning a subtotal is computed. Then any coupons and discounts are entered. A new subtotal is computed and applicable taxes are added. A receipt is printed and the customer pays using cash, credit card, debit card, or check. All appropriate totals (sales, tax, discounts, rebates, etc.) are computed and recorded.

Task Analysis

- Task analysis involves a functional decomposition of tasks to be performed by the system
- Starting at the highest level of abstraction, the designer and customers elicit further levels of detail.
- This detailed decomposition continues until the lowest level of functionality (single task) is achieved.



Partial task analysis for the pet store POS system

Viewpoints

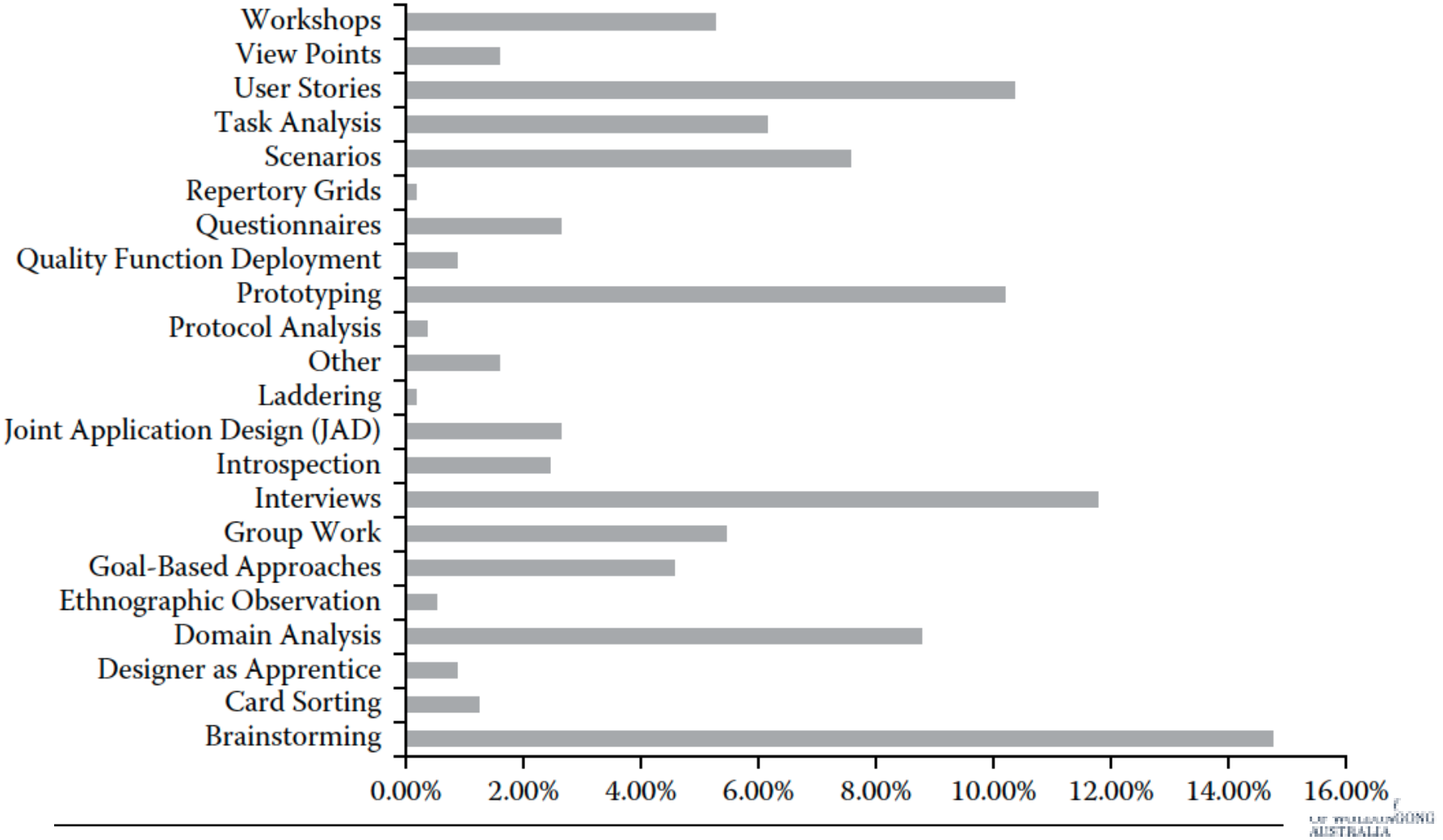
- Viewpoints are a way to organize information from the (point of view of) different parties.
- By recognizing the needs of each of these stakeholders and the contradictions raised by these viewpoints, conflicts can be reconciled using various approaches.
- Sommerville and Sawyer (1997) suggested the following components should be in each viewpoint:
  - A representation style, which defines the notation used in the specification
  - A domain, which is defined as “the area of concern addressed by the viewpoint”
  - A specification, which is a model of a system expressed in the defined style
  - A work plan, with a process model, which defines how to build and check the specification
  - A work record, which is a trace of the actions taken in building, checking, and modifying the specification

Elicitation Techniques Summary

Technique Type	*Techniques*
Domain-oriented	Card sorting Designer as apprentice Domain analysis Laddering Protocol analysis Task analysis
Ethnography	Ethnographic observation
Goals	Goal-based approaches QFD
Group work	Brainstorming Group work JAD Workshops
Interviews	Interviews Introspection Questionnaires
Prototyping	Prototyping
Scenarios	Scenarios Use cases User stories
Viewpoints	Viewpoints Repertory grids

Prevalence of Requirements Elicitation Techniques

Which elicitation techniques are more popular in industry?



Eliciting Hazards

- Hazards are a function of input anomalies that are either naturally occurring or artificially occurring
- These anomalous input events need to be identified and their resultant failure modes and criticality need to be determined during the requirements elicitation phase in order to develop an appropriate set of "shall not" requirements for the system.
- "shall not" behaviors are the set of output behaviors that are undesired and that hazards were a subset of those behaviors that tended to cause serious or catastrophic failures.
- Typical techniques for hazard determination:
  - Misuse cases
  - Antimodeling
  - Formal methods

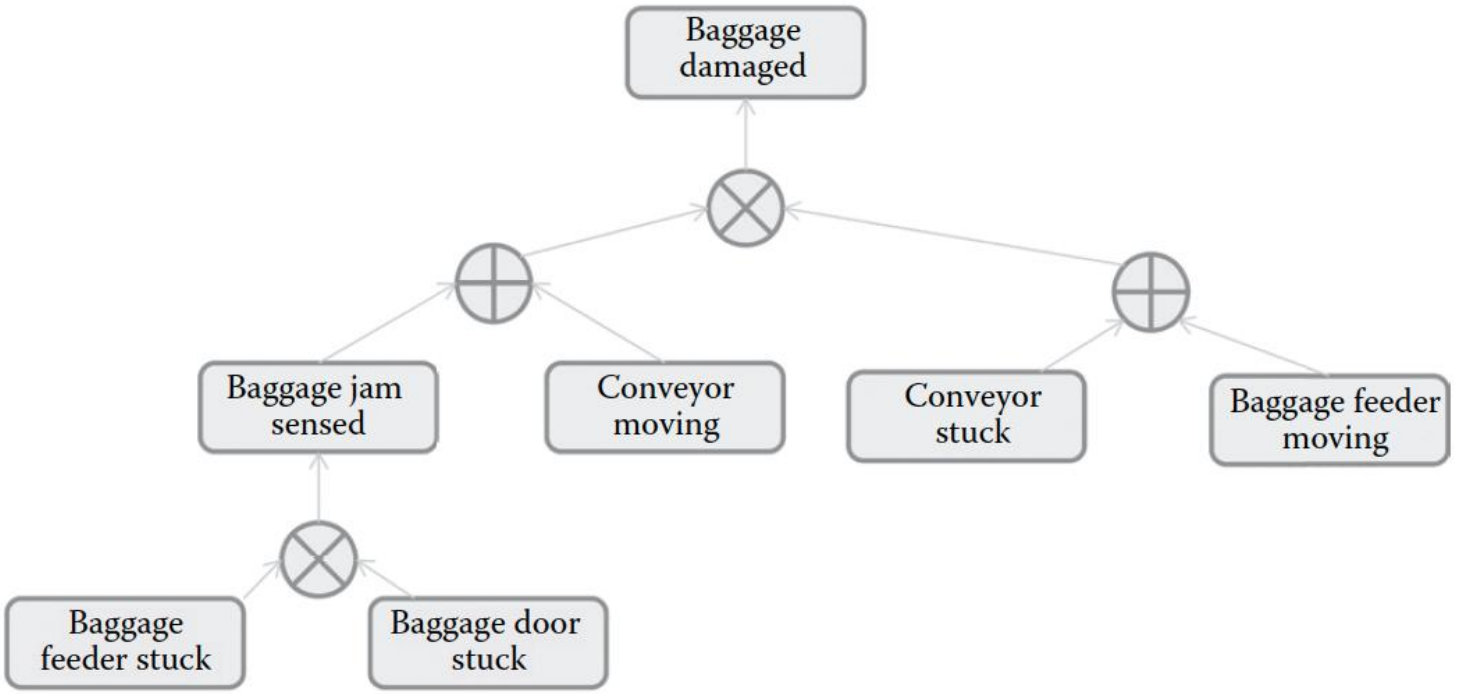
Misuse Case

- Misuse cases (or abuse cases) describe undesired behaviors.
- Typical misuses for most systems include security breaches and other malicious behaviors as well as abuse by untrained, disoriented, or incapable users.
- An easy way to create misuse cases is to assume the role of a persona non grata, that is, an unwanted user of the system, and then model the behaviors of such a person
- For example, in the pet store POS system, it would be appropriate to consider how a hacker would infiltrate this system, and then create requirements that would thwart the hacker's intentions.

Antimodels

- Another way of deriving unwanted behaviour is to create antimodels for the system.
- Antimodels are related to fault trees which are derived by creating a cause and effect hierarchy for unwanted behaviours leading to system failure.
- Then, the causes of the system failure are used to create the “shall not” requirements.

Antimodels



- The fault tree leads us to write the following raw requirements:
  - If a baggage jam is sensed, then the conveyor shall not move.
  - If the baggage feeder is stuck, then the conveyor shall not move.
  - If the baggage door is stuck, then the conveyor shall not move.
  - If the conveyor is stuck, then the baggage feeder should not move.

Formal Methods

- Mathematical formalisms can be used to create a model of the system and its environment as related to their goals, operations, requirements, and constraints.
- These formalisms can then be used in conjunction with automated model checkers to examine various properties of the system and ensure that unwanted ones are not present.
- The uniquely identified hazards can be listed in a special section in the requirements document, or be designated by a flag beside the requirement, or be tagged within a requirements management tool.