U O W

Software Requirements, Specifications and Formal Methods

Lecture 02 – Requirements Engineering Processes

Slides mainly taken from **Dr. Fenghui Ren** of UOW

UNIVERSITY OF WOLLONGONG AUSTRALIA

Recap on Lecture 01

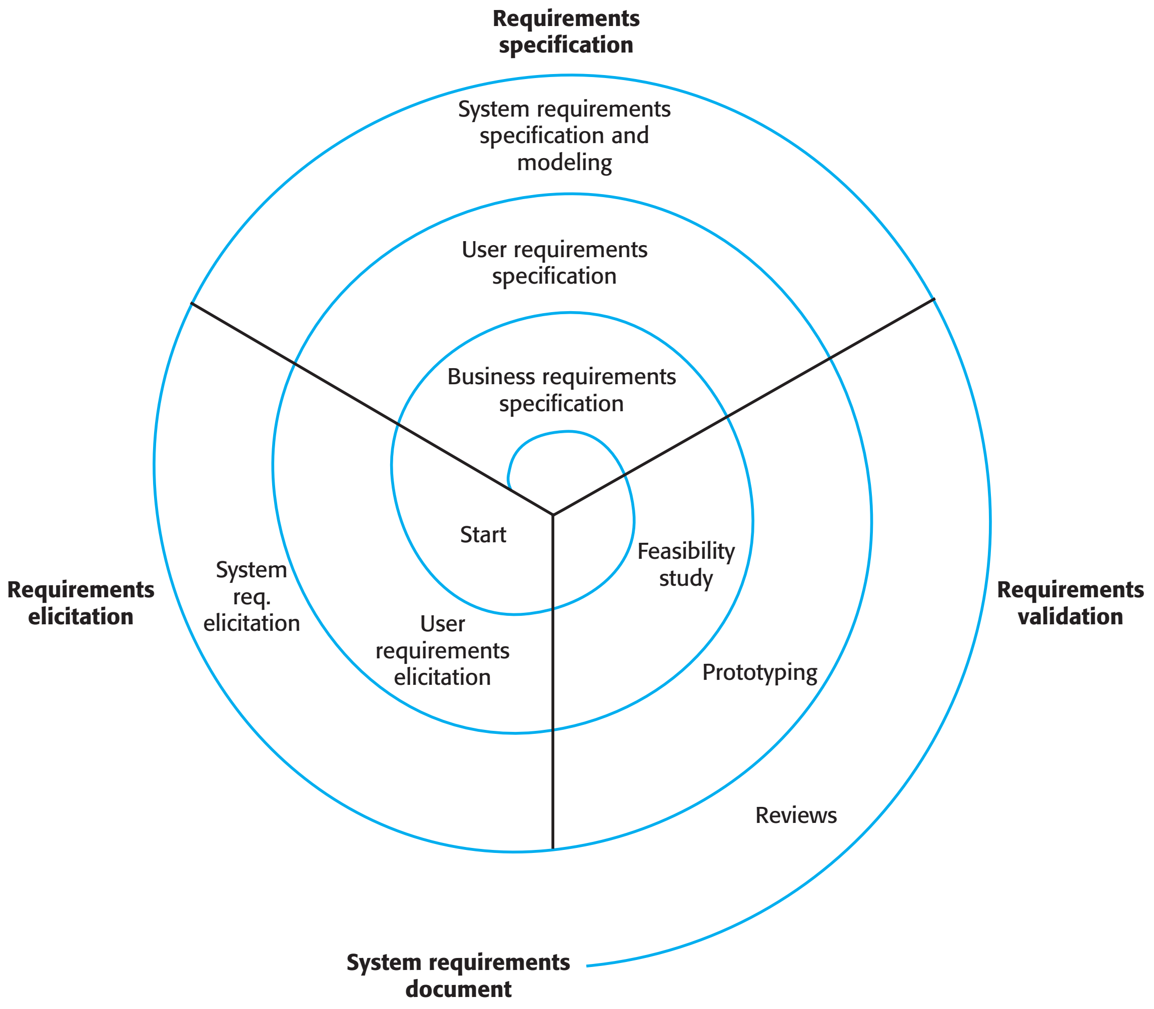
- Software Engineering
- Software Process Models
  - Waterfall model
  - Incremental development
  - Integration and configuration
- Requirements Engineering
- Types of Requirements
  - User requirements & system requirements
  - Functional requirements & non-functional requirements
- Requirement Documents

UNIVERSITY OF WOLLONGONG AUSTRALIA

Requirements engineering processes

- ✧ The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements.
- ✧ However, there are a number of generic activities common to all processes
  - Requirements elicitation;
  - Requirements analysis;
  - Requirements validation;
  - Requirements management.
- ✧ In practice, RE is an iterative activity in which these processes are interleaved.

A spiral view of the requirements engineering process


Requirements specification
System requirements specification and modeling
User requirements specification
Business requirements specification
Start
Feasibility study
Requirements elicitation
System req. elicitation
User requirements elicitation
Prototyping
Requirements validation
Reviews
System requirements document

Requirements elicitation and analysis

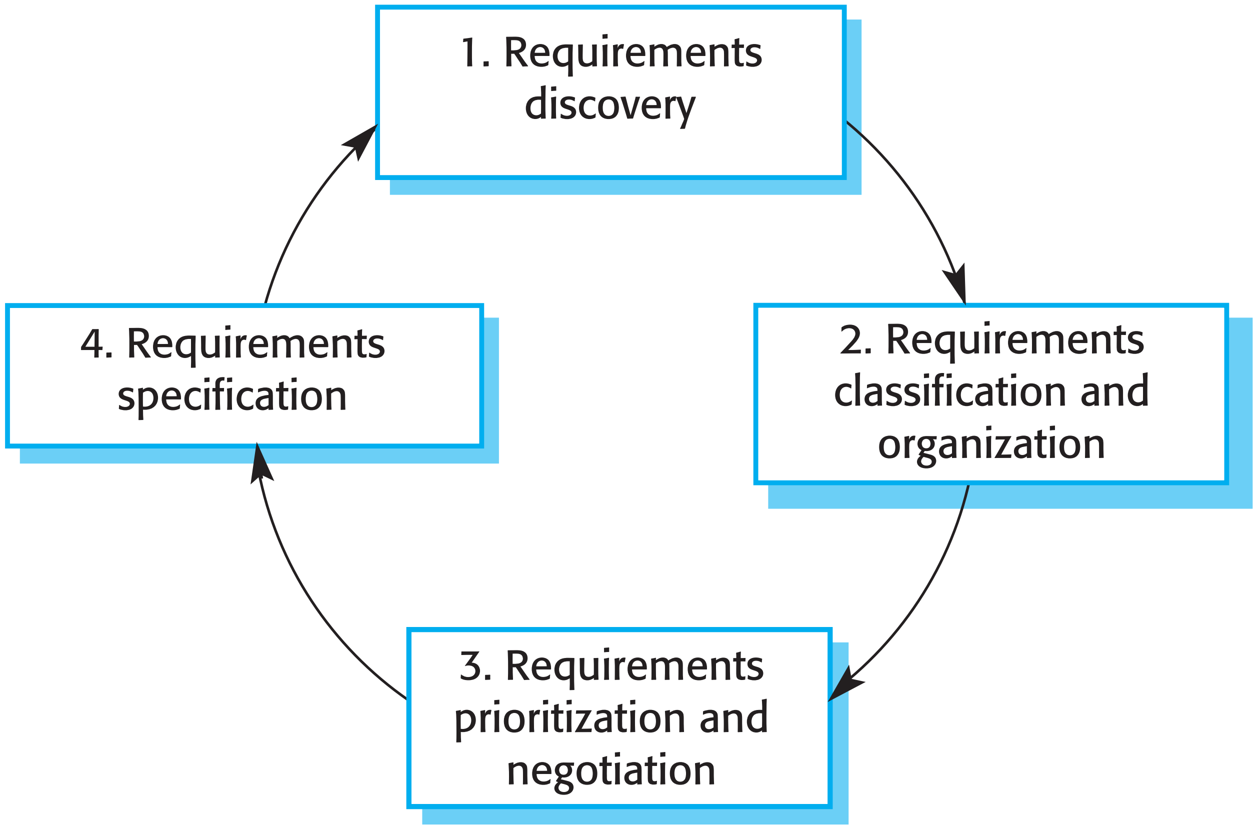
- Sometimes called requirements elicitation or requirements discovery.
- Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called *stakeholders.*

UNIVERSITY OF WOLLONGONG AUSTRALIA

Problems of requirements elicitation

- ✧ Stakeholders don’t know what they really want.
- ✧ Stakeholders express requirements in their own terms.
- ✧ Different stakeholders may have conflicting requirements.
- ✧ Organisational and political factors may influence the system requirements.
- ✧ The requirements change during the analysis process. New stakeholders may emerge and the business environment may change.

The requirements elicitation and analysis process


1. Requirements discovery
2. Requirements classification and organization
3. Requirements prioritization and negotiation
4. Requirements specification

Process activities

- Requirements discovery
  - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
- Requirements classification and organisation
  - Groups related requirements and organises them into coherent clusters.
- Prioritisation and negotiation
  - Prioritising requirements and resolving requirements conflicts.
- Requirements specification
  - Requirements are documented and input into the next round of the spiral.

Requirements discovery

- The process of gathering information about the required and existing systems and distilling the user and system requirements from this information.
- Interaction is with system stakeholders from managers to external regulators.
- Systems normally have a range of stakeholders.

Interviews in practice

- ✧ Normally a mix of closed and open-ended interviewing.
- ✧ Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.
- ✧ Interviewers need to be open-minded without pre-conceived ideas of what the system should do
- ✧ Interviewers need to prompt the interviewee to talk about the system by suggesting requirements rather than simply asking them what they want.

UNIVERSITY OF WOLLONGONG AUSTRALIA

Problems with interviews

- ✧ Application specialists may use language to describe their work that isn't easy for the requirements engineer to understand.
- ✧ Interviews are not good for understanding domain requirements
  - Requirements engineers cannot understand specific domain terminology;
  - Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating.

Stories and scenarios

✧ Scenarios and user stories are real-life examples of how a system can be used.

✧ Stories and scenarios are a description of how a system may be used for a particular task.

✧ Because they are based on a practical situation, stakeholders can relate to them and can comment on their situation with respect to the story.

Scenarios

- A structured form of user story
- Scenarios should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.

UNIVERSITY OF WOLLONGONG AUSTRALIA

Requirements specification

- ✧ The process of writing down the user and system requirements in a requirements document.
- ✧ User requirements have to be understandable by end-users and customers who do not have a technical background.
- ✧ System requirements are more detailed requirements and may include more technical information.
- ✧ The requirements may be part of a contract for the system development
  - It is therefore important that these are as complete as possible.

Ways of writing a system requirements specification

Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract

Requirements and design

- In principle, requirements should state what the system should do and the design should describe how it does this.
- In practice, requirements and design are inseparable
  - A system architecture may be designed to structure the requirements;
  - The system may inter-operate with other systems that generate design requirements;
  - The use of a specific architecture to satisfy non-functional requirements may be a domain requirement.
  - This may be the consequence of a regulatory requirement.

UNIVERSITY OF WOLLONGONG AUSTRALIA

Natural language specification

- ✧ Requirements are written as natural language sentences supplemented by diagrams and tables.
- ✧ Used for writing requirements because it is expressive, intuitive and universal. This means that the requirements can be understood by users and customers.

UNIVERSITY OF WOLLONGONG AUSTRALIA

Guidelines for writing requirements

- ✧ Invent a standard format and use it for all requirements.
- ✧ Use language in a consistent way. Use shall for mandatory requirements, should for desirable requirements.
- ✧ Use text highlighting to identify key parts of the requirement.
- ✧ Avoid the use of computer jargon.
- ✧ Include an explanation (rationale) of why a requirement is necessary.

Problems with natural language

- ✧ Lack of clarity
  - Precision is difficult without making the document difficult to read.
- ✧ Requirements confusion
  - Functional and non-functional requirements tend to be mixed-up.
- ✧ Requirements amalgamation
  - Several different requirements may be expressed together.

Structured specifications

- ✧ An approach to writing requirements where the freedom of the requirements writer is limited and requirements are written in a standard way.
- ✧ This works well for some types of requirements e.g. requirements for embedded control system but is sometimes too rigid for writing business system requirements.

Form-based specifications

- ✧ Definition of the function or entity.
- ✧ Description of inputs and where they come from.
- ✧ Description of outputs and where they go to.
- ✧ Information about the information needed for the computation and other entities used.
- ✧ Description of the action to be taken.
- ✧ Pre and post conditions (if appropriate).
- ✧ The side effects (if any) of the function.

Tabular specification

- ✧ Used to supplement natural language.
- ✧ Particularly useful when you have to define a number of possible alternative courses of action.
- ✧ For example, the insulin pump systems bases its computations on the rate of change of blood sugar level and the tabular specification explains how to calculate the insulin requirement for different scenarios.

Model-based Specification

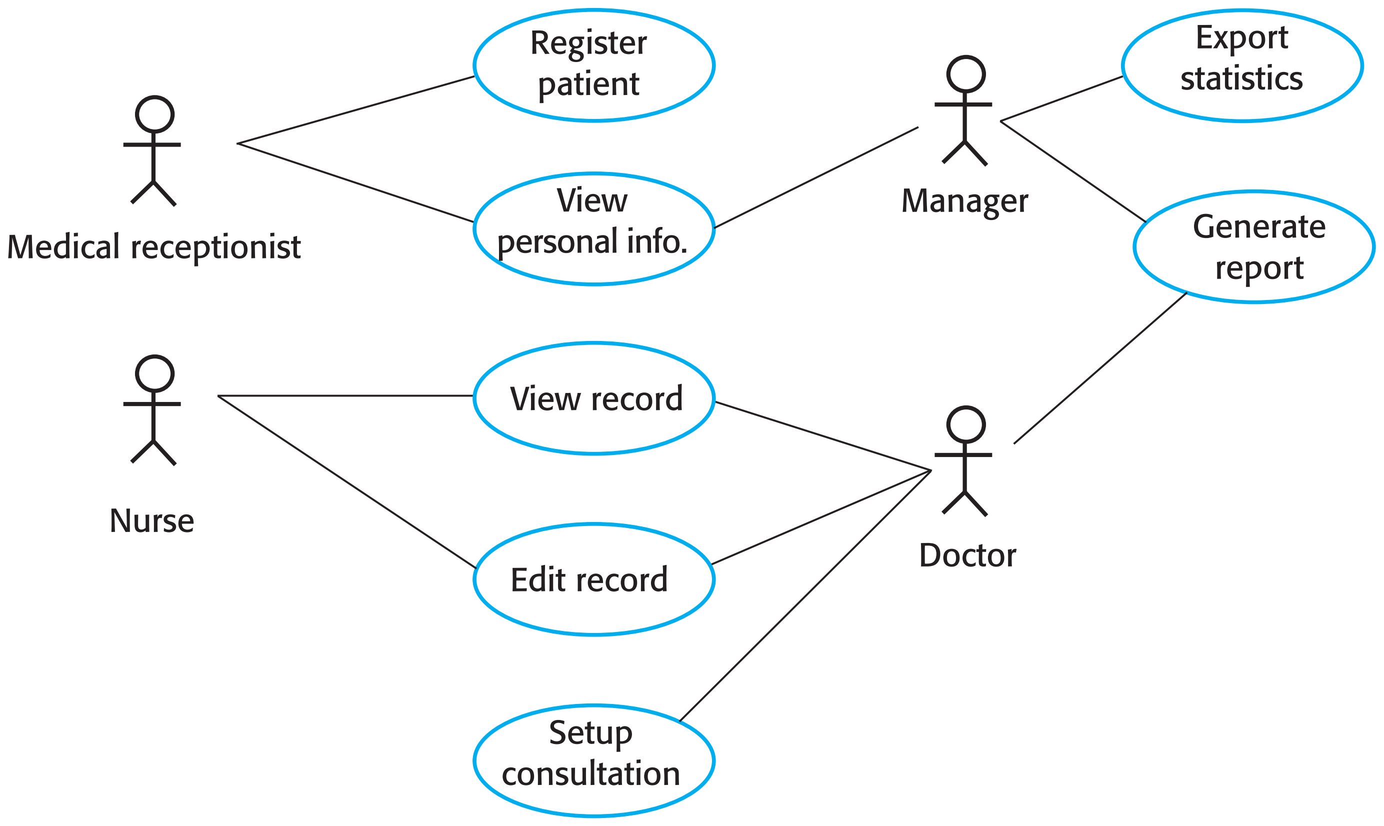
- ✧ Formal specification
- ✧ Mathematical system model
- ✧ Z specification language

Use cases

- Use-cases are a kind of scenario that are included in the UML.
- Use cases identify the actors in an interaction and which describe the interaction itself.
- A set of use cases should describe all possible interactions with the system.
- High-level graphical model supplemented by more detailed tabular description
- UML sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

UNIVERSITY OF WOLLONGONG AUSTRALIA

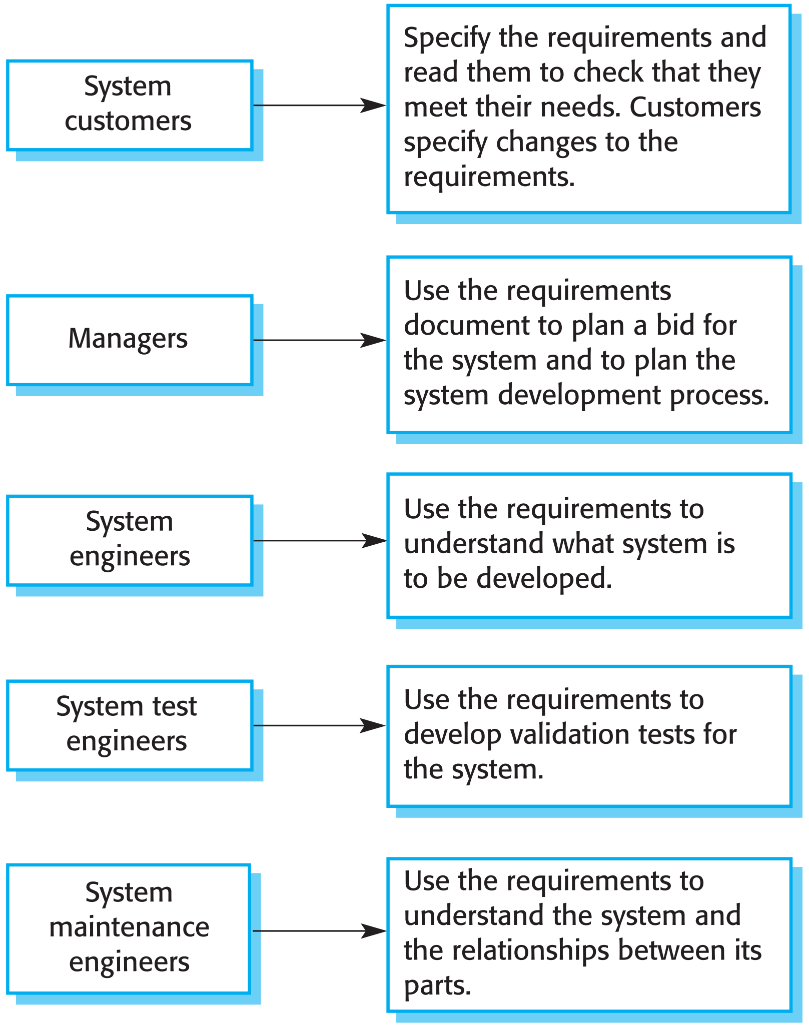
Use cases for the Mentcare system



The software requirements document

- ✧ The software requirements document is the official statement of what is required of the system developers.
- ✧ Should include both a definition of user requirements and a specification of the system requirements.
- ✧ It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it.

Users of a requirements document


System customers
Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
Managers
Use the requirements document to plan a bid for the system and to plan the system development process.
System engineers
Use the requirements to understand what system is to be developed.
System test engineers
Use the requirements to develop validation tests for the system.
System maintenance engineers
Use the requirements to understand the system and the relationships between its parts.

UNIVERSITY OF WOLLONGONG AUSTRALIA

Requirements document variability

- ✧ Information in requirements document depends on type of system and the approach to development used.
- ✧ Systems developed incrementally will, typically, have less detail in the requirements document.
- ✧ Requirements documents standards have been designed e.g. IEEE standard. These are mostly applicable to the requirements for large systems engineering projects.

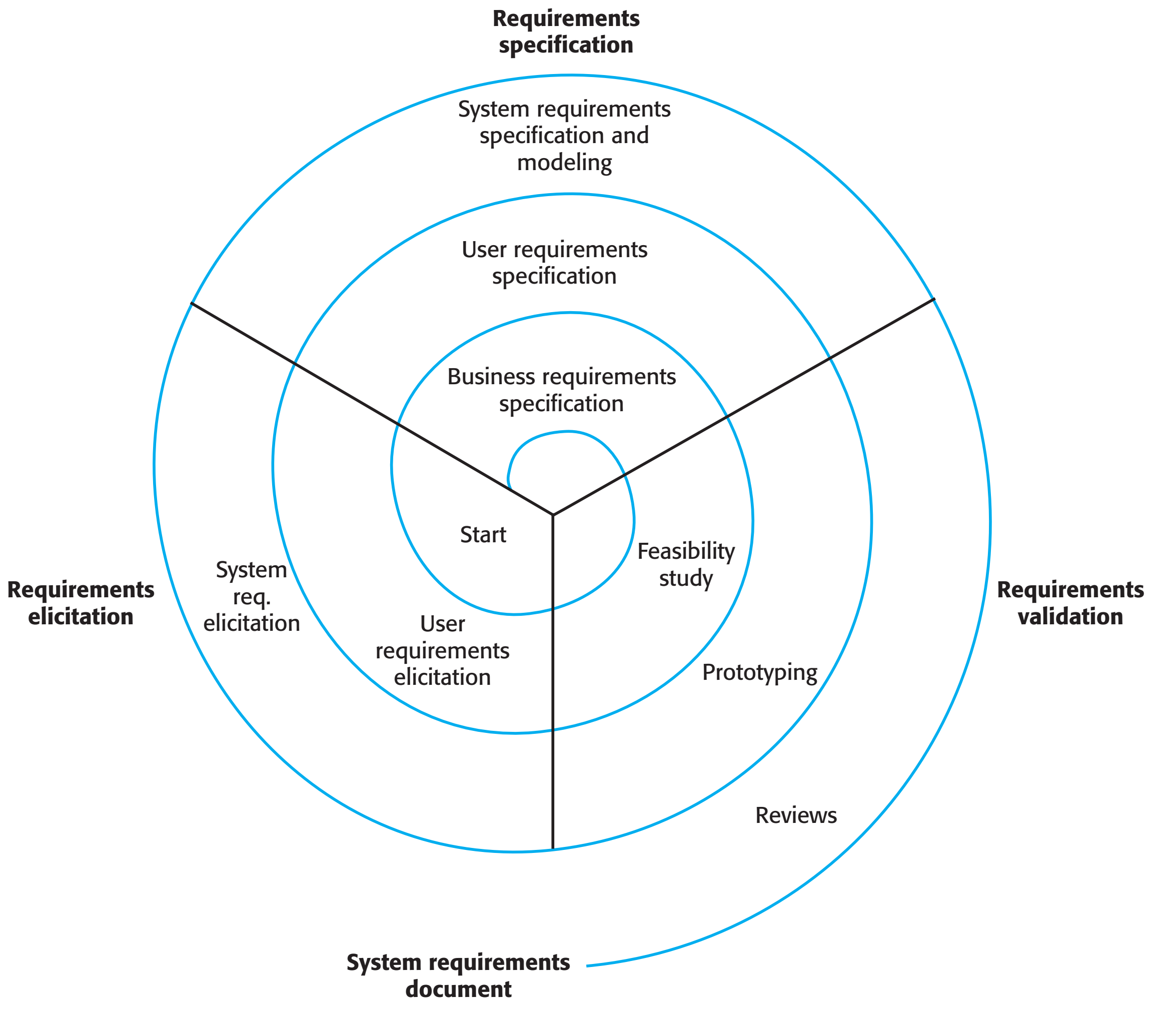
The structure of a requirements document

Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.

The structure of a requirements document

Chapter	Description
System requirements specification	This should describe the functional and nonfunctional requirements in more detail. If necessary, further detail may also be added to the nonfunctional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components and the system and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.

A spiral view of the requirements engineering process


Requirements specification
System requirements specification and modeling
User requirements specification
Business requirements specification
Start
Feasibility study
Requirements elicitation
System req. elicitation
User requirements elicitation
Prototyping
Requirements validation
Reviews
System requirements document

UNIVERSITY OF WOLLONGONG AUSTRALIA

Requirements validation

- ✧ Concerned with demonstrating that the requirements define the system that the customer really wants.
- ✧ Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

Requirements checking

- ✧ **Validity**. Does the system provide the functions which best support the customer's needs?
- ✧ **Consistency**. Are there any requirements conflicts?
- ✧ **Completeness**. Are all functions required by the customer included?
- ✧ **Realism**. Can the requirements be implemented given available budget and technology
- ✧ **Verifiability**. Can the requirements be checked?

Requirements validation techniques

- Requirements reviews
  - Systematic manual analysis of the requirements.
- Prototyping
  - Using an executable model of the system to check requirements.
- Test-case generation
  - Developing tests for requirements to check testability.

Software prototyping

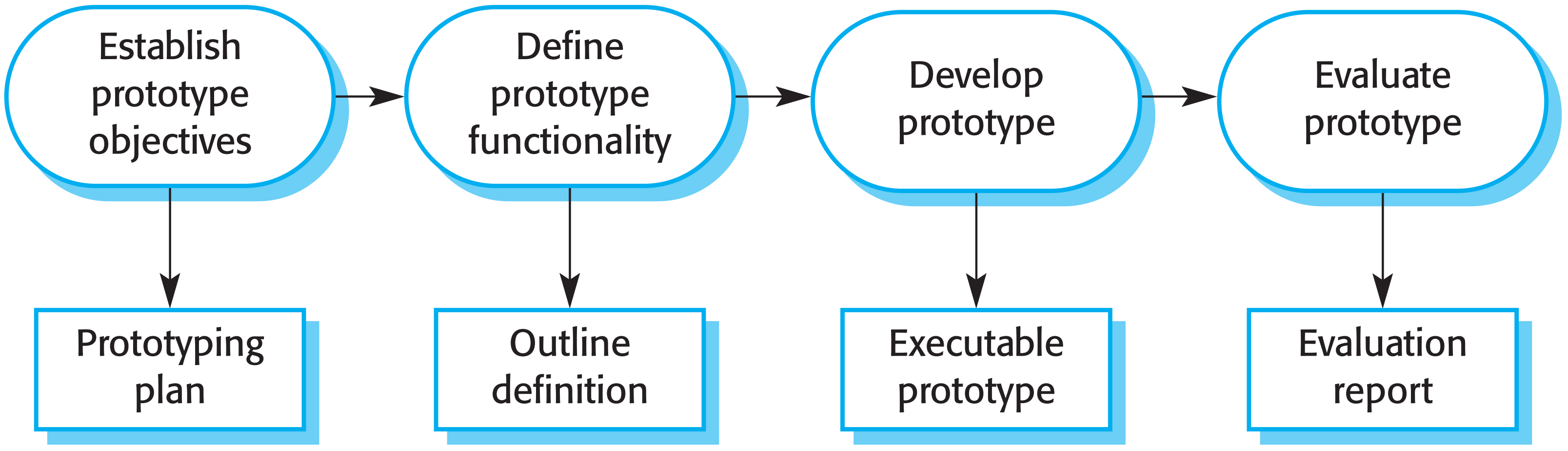
- ✧ A prototype is an initial version of a system used to demonstrate concepts and try out design options.
- ✧ A prototype can be used in:
  - The requirements engineering process to help with requirements elicitation and validation;
  - In design processes to explore options and develop a UI design;
  - In the testing process to run back-to-back tests.

Benefits of prototyping

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

UNIVERSITY OF WOLLONGONG AUSTRALIA

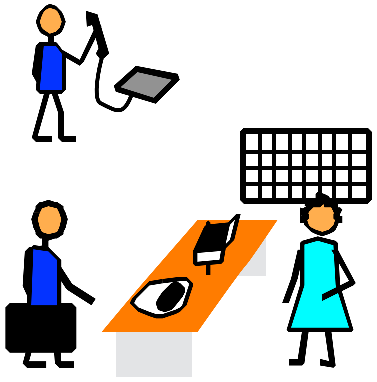
The process of prototype development

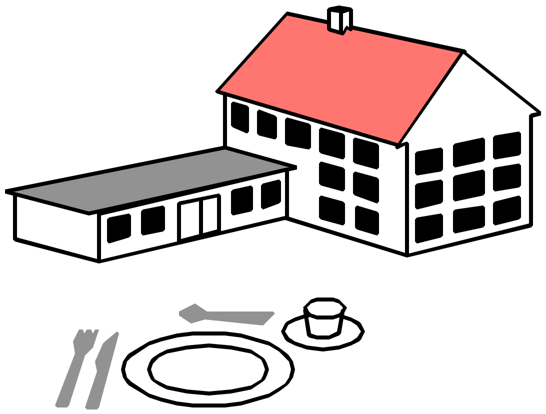


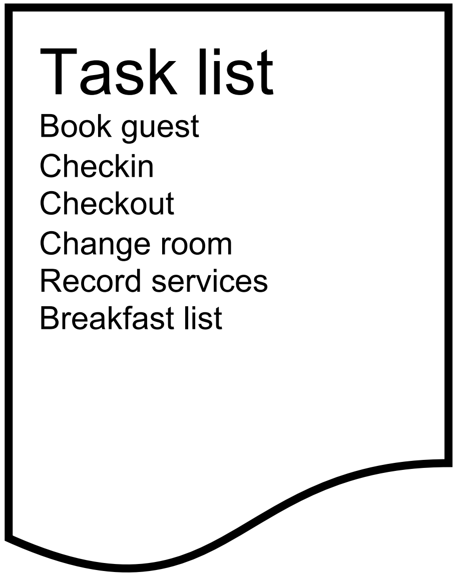
Prototype development

- 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

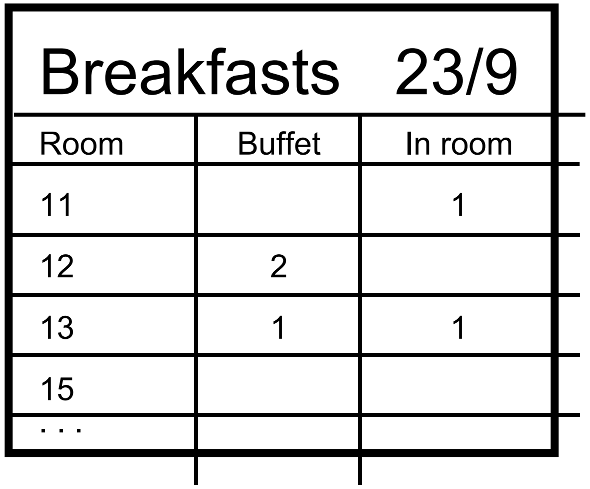






Task list

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*.

User tasks (continued)

When the guest receives breakfast, makes phone calls or receives other services, the system must record it and put it on the bill. This is the *record-service task.*

Many hotels have a restaurant where the guests can get breakfast, and the staff use a special list for recording who got what. The list has one line per hotel room. The guest can have two types of breakfast, i.e. breakfast served in the room and breakfast served buffet style in the restaurant. This is the *breakfast-list task*.

The first Hotel System Prototype

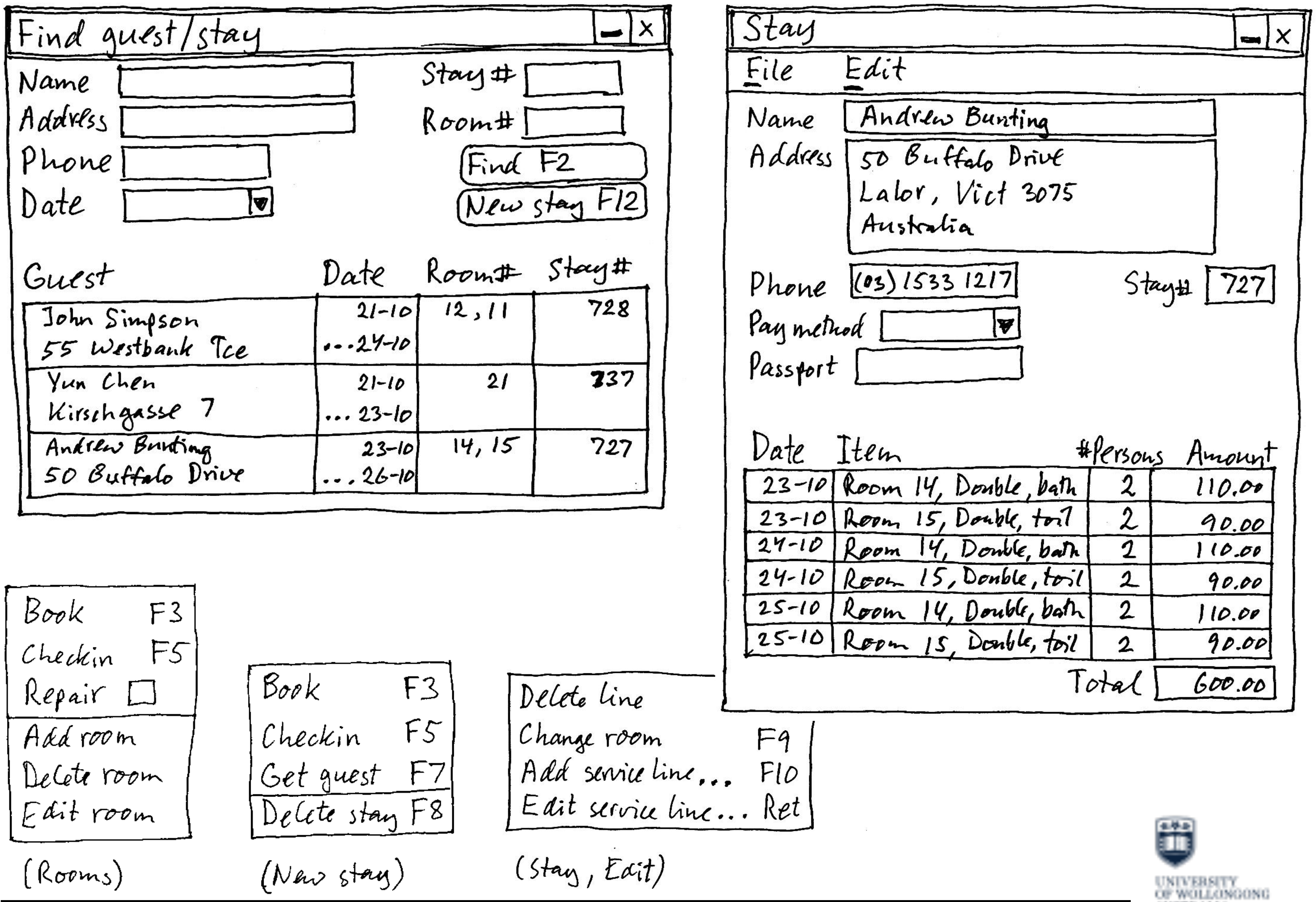
We will look at the first prototype of the hotel system and how was its usability tested and what usability problems we found.

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

Hotel system prototype

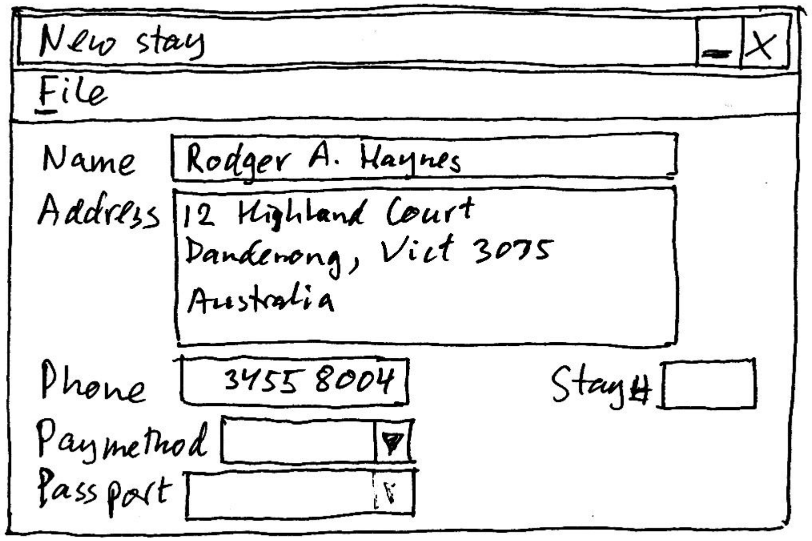


Guest	Date	Room#	Stay#
John Simpson 55 Westbank Tce	21-10 ...24-10	12, 11	728
Yun Chen Kirschgasse 7	21-10 ... 23-10	21	737
Andrew Bunting 50 Buffalo Drive	23-10 ... 26-10	14, 15	727

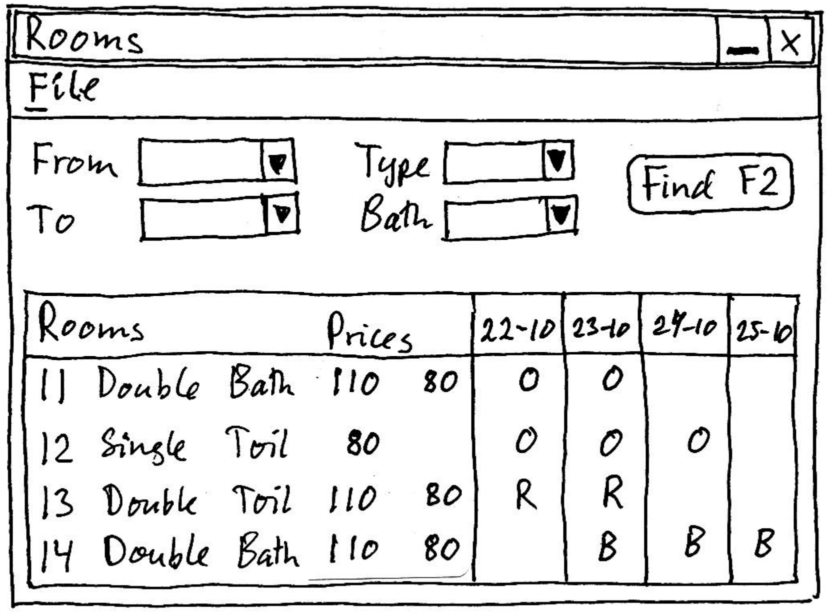
Name: Andrew Bunting
Address: 50 Buffalo Drive
Lalor, Vict 3075
Australia
Phone: (03) 1533 1217
Stay#: 727
Pay method:
Passport:

Date	Item	#Persons	Amount
23-10	Room 14, Double, bath	2	110.00
23-10	Room 15, Double, toil	2	90.00
24-10	Room 14, Double, bath	2	110.00
24-10	Room 15, Double, toil	2	90.00
25-10	Room 14, Double, bath	2	110.00
25-10	Room 15, Double, toil	2	90.00

Total: 600.00



Name	Rodger A. Haynes
Address	12 Highland Court Dandenong, Vict 3075 Australia
Phone	3455 8004
Stay#	
Paymethod	
Passport	



Rooms	Prices		22-10	23-10	24-10	25-10
11 Double Bath	110	80	O	O		
12 Single Toil	80		O	O	O	
13 Double Toil	110	80	R	R		
14 Double Bath	110	80		B	B	B

Types of Prototypes

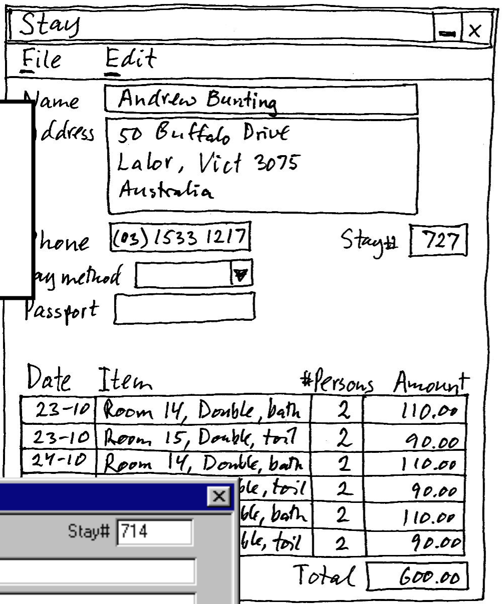
- ✧ **Hand-draw mock-up.** The designer draws the screens by hand using paper and pencil.
- ✧ **Tool-draw mock-up.** The designer draws the screens on the computer using the same tool that will be used in the final product.
- ✧ **Screen prototype.** The screens are shown on the real computer screen,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

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 detect 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.

Various prototypes



Stay

File Edit

ame: Andrew Bunting

ddress: 50 Buffalo Drive
Lalor, Vict 3075
Australia

hone: (03) 1533 1217 Stay# 727

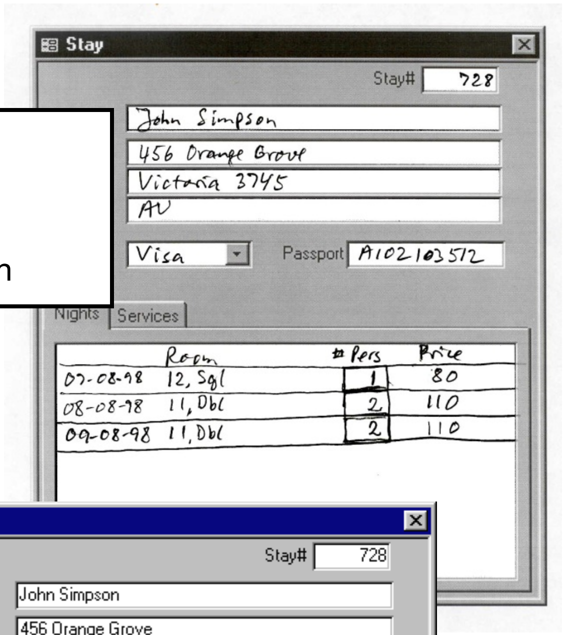
ay method:

Passport:

Date	Item	#Persons	Amount
23-10	Room 14, Double, bath	2	110.00
23-10	Room 15, Double, toil	2	90.00
24-10	Room 14, Double, bath	2	110.00
	ble, toil	2	90.00
	ble, bath	2	110.00
	ble, toil	2	90.00
		Total	600.00

Hand-drawn mockup:

15-30 min



Stay

Stay# 728

John Simpson
456 Orange Grove
Victoria 3745
AU

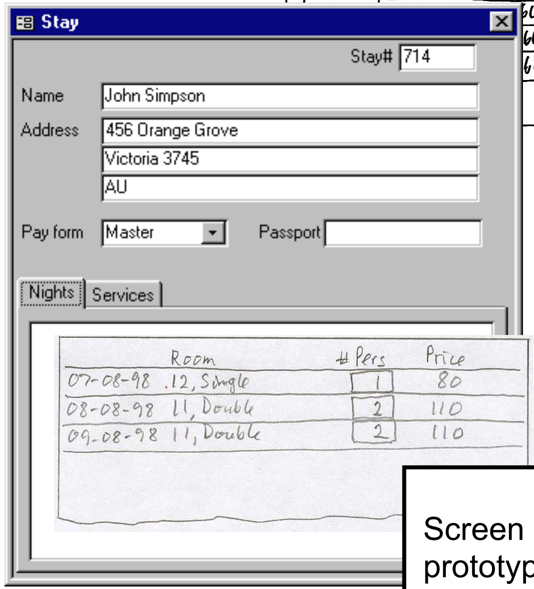
Visa Passport A102 103 512

Nights Services

	Room	# Pers	Price
07-08-98	12, Sgl	1	80
08-08-98	11, Dbl	2	110
09-08-98	11, Dbl	2	110

Tool-drawn mockup:

30-60 min



Stay

Stay# 714

Name: John Simpson

Address: 456 Orange Grove
Victoria 3745
AU

Pay form: Master Passport:

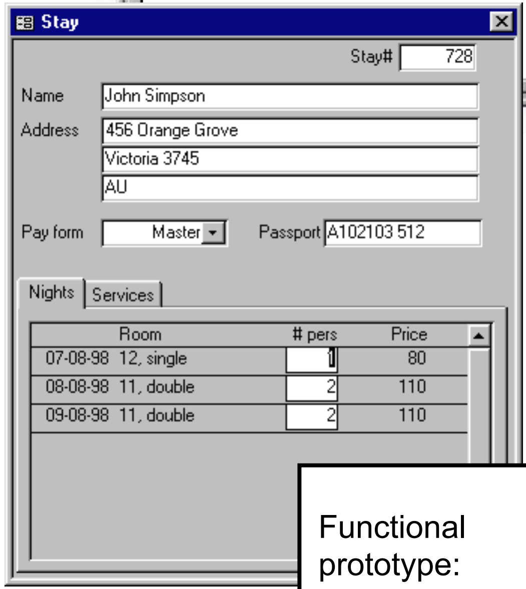
Nights Services

	Room	# Pers	Price
07-08-98	12, Single	1	80
08-08-98	11, Double	2	110
09-08-98	11, Double	2	110

Screen prototype:

1-4 hours

Which prototype is the best?



Stay

Stay# 728

Name: John Simpson

Address: 456 Orange Grove
Victoria 3745
AU

Pay form: Master Passport: A102103 512

Nights Services

	Room	# pers	Price
07-08-98	12, single	1	80
08-08-98	11, double	2	110
09-08-98	11, double	2	110

Functional prototype:

2-8 hours

Changing requirements

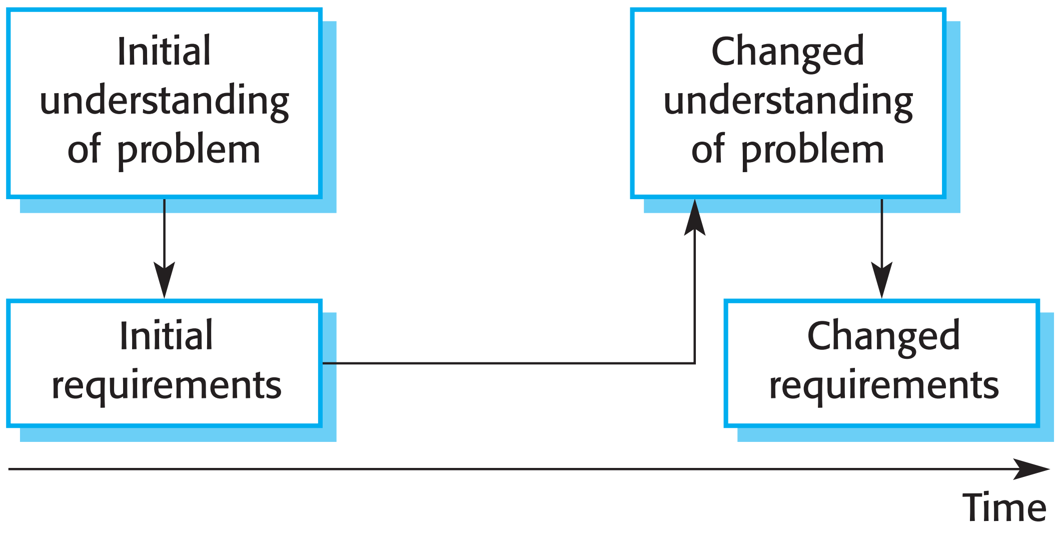
- ✧ The business and technical environment of the system always changes after installation.
  - New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change (with consequent changes in the system support required), and new legislation and regulations may be introduced that the system must necessarily abide by.
- ✧ The people who pay for a system and the users of that system are rarely the same people.
  - System customers impose requirements because of organizational and budgetary constraints. These may conflict with end-user requirements and, after delivery, new features may have to be added for user support if the system is to meet its goals.

Changing requirements

- ✧ Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.
  - The final system requirements are inevitably a compromise between them and, with experience, it is often discovered that the balance of support given to different users has to be changed.

UNIVERSITY OF WOLLONGONG AUSTRALIA

Requirements evolution


Initial understanding of problem
Changed understanding of problem
Initial requirements
Changed requirements
Time

Requirements management

- ✧ Requirements management is the process of managing changing requirements during the requirements engineering process and system development.
- ✧ New requirements emerge as a system is being developed and after it has gone into use.
- ✧ You need to keep track of individual requirements and maintain links between dependent requirements so that you can assess the impact of requirements changes. You need to establish a formal process for making change proposals and linking these to system requirements.

UNIVERSITY OF WOLLONGONG AUSTRALIA

Requirements management planning

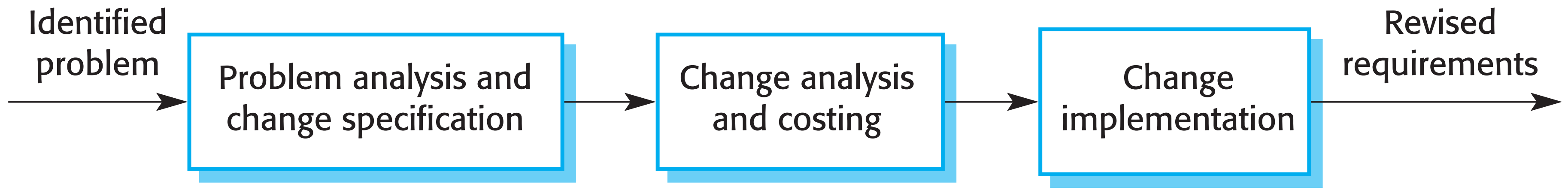
- ✧ Establishes the level of requirements management detail that is required.
- ✧ Requirements management decisions:
  - *Requirements identification* Each requirement must be uniquely identified so that it can be cross-referenced with other requirements.
  - *A change management process* This is the set of activities that assess the impact and cost of changes.
  - *Traceability policies* These policies define the relationships between each requirement and between the requirements and the system design that should be recorded.
  - *Tool support* Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems.

UNIVERSITY OF WOLLONGONG AUSTRALIA

Requirements change management

- Deciding if a requirements change should be accepted
  - *Problem analysis and change specification*
    - During this stage, the problem or the change proposal is analyzed to check that it is valid. This analysis is fed back to the change requestor who may respond with a more specific requirements change proposal, or decide to withdraw the request.
  - *Change analysis and costing*
    - The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. Once this analysis is completed, a decision is made whether or not to proceed with the requirements change.
  - Change implementation
    - The requirements document and, where necessary, the system design and implementation, are modified. Ideally, the document should be organized so that changes can be easily implemented.

Requirements change management


Identified problem
Problem analysis and change specification
Change analysis and costing
Change implementation
Revised requirements