Requirements and Requirements Specification.

Topics covered

✧ Functional and non-functional requirements

What is a requirement?

- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.

Types of requirement

- ✧ User requirements
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- ✧ System requirements
  - A structured document setting out detailed descriptions of the system’s functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

User and system requirements

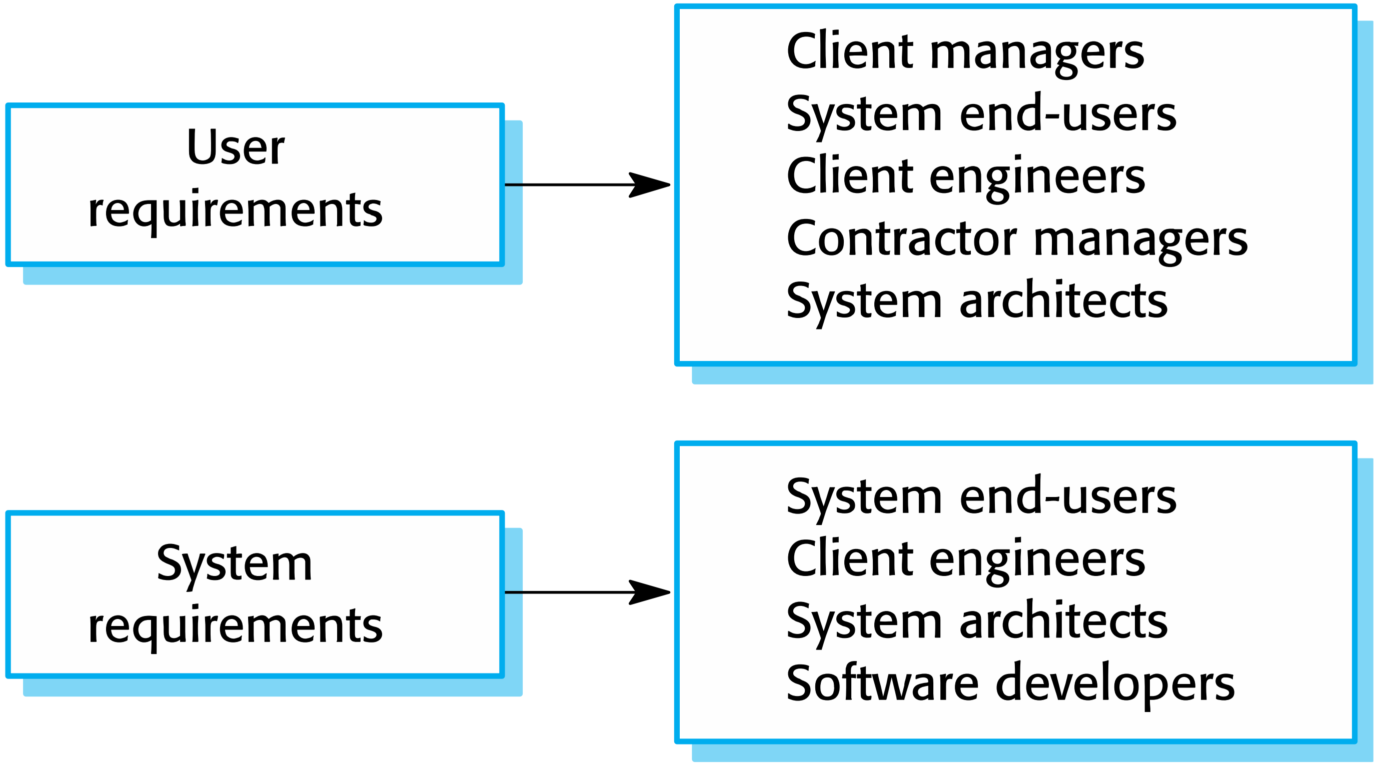
User requirements definition

1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20mg, etc) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

Readers of different types of requirements specification



System stakeholders

- ✧ Any person or organization who is affected by the system in some way and so who has a legitimate interest
- ✧ Stakeholder types
  - End users
  - System managers
  - System owners
  - External stakeholders

Stakeholders in the Mentcare system

- ✧ Patients whose information is recorded in the system.
- ✧ Doctors who are responsible for assessing and treating patients.
- ✧ Nurses who coordinate the consultations with doctors and administer some treatments.
- ✧ Medical receptionists who manage patients' appointments.
- ✧ IT staff who are responsible for installing and maintaining the system.

Stakeholders in the Mentcare system

- ✧ A medical ethics manager who must ensure that the system meets current ethical guidelines for patient care.
- ✧ Health care managers who obtain management information from the system.
- ✧ Medical records staff who are responsible for ensuring that system information can be maintained and preserved, and that record keeping procedures have been properly implemented.

Agile methods and requirements

- ✧ Many agile methods argue that producing detailed system requirements is a waste of time as requirements change so quickly.
- ✧ The requirements document is therefore always out of date.
- ✧ Agile methods usually use incremental requirements engineering and may express requirements as ‘user stories’ (discussed in Chapter 3).
- ✧ This is practical for business systems but problematic for systems that require pre-delivery analysis (e.g. critical systems) or systems developed by several teams.

Functional and non-functional requirements

Functional and non-functional requirements

- ✧ Functional requirements
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
  - May state what the system should not do.
- ✧ Non-functional requirements
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a whole rather than individual features or services.
- ✧ Domain requirements
  - Constraints on the system from the domain of operation

Functional requirements

- ✧ Describe functionality or system services.
- ✧ Depend on the type of software, expected users and the type of system where the software is used.
- ✧ Functional user requirements may be high-level statements of what the system should do.
- ✧ Functional system requirements should describe the system services in detail.

Mentcare system: functional requirements

- ✧ A user shall be able to search the appointments lists for all clinics.
- ✧ The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
- ✧ Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Requirements imprecision

- ✧ Problems arise when functional requirements are not precisely stated.
- ✧ Ambiguous requirements may be interpreted in different ways by developers and users.
- ✧ Consider the term ‘search’ in requirement 1
  - User intention – search for a patient name across all appointments in all clinics;
  - Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.

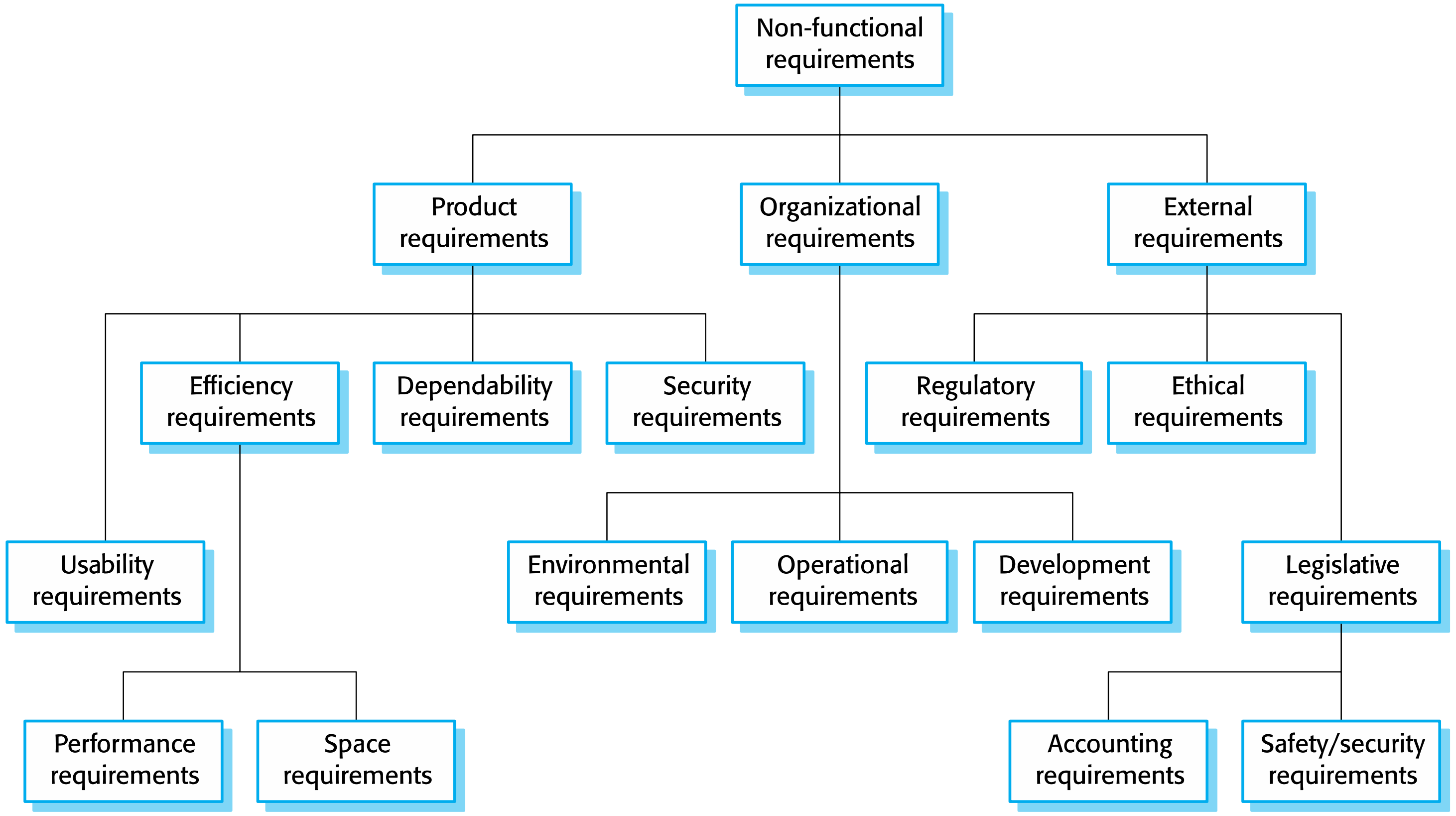
Requirements completeness and consistency

- In principle, requirements should be both complete and consistent.
- Complete
  - They should include descriptions of all facilities required.
- Consistent
  - There should be no conflicts or contradictions in the descriptions of the system facilities.
- In practice, because of system and environmental complexity, it is impossible to produce a complete and consistent requirements document.

Non-functional requirements

- These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular IDE, programming language or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

Types of nonfunctional requirement



Non-functional requirements implementation

- ✧ Non-functional requirements may affect the overall architecture of a system rather than the individual components.
  - ▪ For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- ✧ A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
  - ▪ It may also generate requirements that restrict existing requirements.

Non-functional classifications

- ✧ Product requirements
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- ✧ Organisational requirements
  - Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- ✧ External requirements
  - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Examples of nonfunctional requirements in the Mentcare system

Product requirement
The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

Organizational requirement
Users of the Mentcare system shall authenticate themselves using their health authority identity card.

External requirement
The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.

Usability requirements

- ✧ The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized. (Goal)
- ✧ Medical staff shall be able to use all the system functions after four hours of training. After this training, the average number of errors made by experienced users shall not exceed two per hour of system use. (Testable non-functional requirement)

Metrics for specifying nonfunctional requirements

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Documenting Requirements to a Standard

Documentation Elements

- ✧ A Software Requirement Specification (SRS) document contains both User and System Requirements.
  - In Agile, the requirement may not be complete for the majority of the lifetime of the project. But SRS needs to be as much comprehensive as possible at any stage.
- ✧ Requirements are ‘engineered’ using various methods such interviews, one-to-one and group discussions.
- ✧ Gathered information is documented using various formats/templates:
  - Personas
  - Scenarios
  - User Stories
  - Features

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.

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.

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.

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.

Example requirements for the insulin pump software system

3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. *(Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)*

3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1. *(A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible.)*

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.

A structured specification of a requirement for an insulin pump

Insulin Pump/Control Software/SRS/3.3.2

Function Compute insulin dose: safe sugar level.

Description

Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.

Inputs Current sugar reading (r2); the previous two readings (r0 and r1).

Source Current sugar reading from sensor. Other readings from memory.

Outputs CompDose—the dose in insulin to be delivered.

Destination Main control loop.

A structured specification of a requirement for an insulin pump

Action

CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.

Requirements

Two previous readings so that the rate of change of sugar level can be computed.

Pre-condition

The insulin reservoir contains at least the maximum allowed single dose of insulin.

Post-condition r0 is replaced by r1 then r1 is replaced by r2.

Side effects None.

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.

Tabular specification of computation for an insulin pump

Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase decreasing ((r2 – r1) < (r1 – r0))	CompDose = 0
Sugar level increasing and rate of increase stable or increasing ((r2 – r1) ≥ (r1 – r0))	CompDose = round ((r2 – r1)/4) If rounded result = 0 then CompDose = MinimumDose

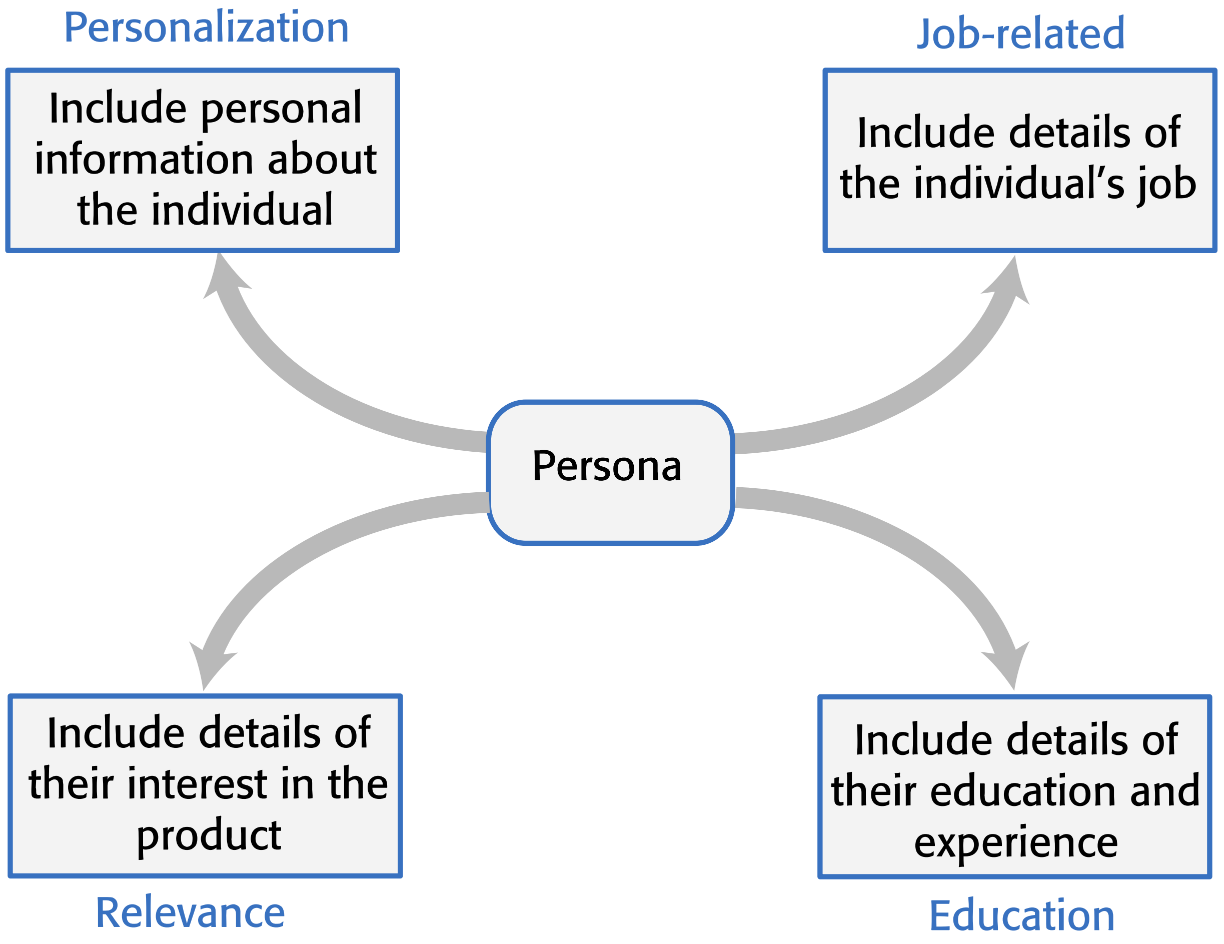
Personas

- ✧ Personas are used to build an understanding of potential users to design features that they are likely to find useful and to design a user interface that is suited to them.
- ✧ Personas are 'imagined users' where you create a character portrait of a type of user that you think might use your product.
  - ▪ For example, if your product is aimed at managing appointments for dentists, you might create a dentist persona, a receptionist persona and a patient persona.
- ✧ Personas of different types of user help you imagine what these users may want to do with your software and how it might be used.

Personas Description

- ✧ A persona should ‘paint a picture’ of a type of product user. They should be relatively short and easy-to-read.
- ✧ You should describe their background and why they might want to use your product.
- ✧ You should also say something about their educational background and technical skills.
- ✧ These help you assess whether or not a software feature is likely to be useful, understandable and usable by typical product users

Personas Description


Personalization
Include personal information about the individual
Job-related
Include details of the individual's job
Persona
Include details of their interest in the product
Relevance
Include details of their education and experience
Education

Persona (Example)

Jack, a primary school teacher

Jack, age 32, is a primary school (elementary school) teacher in Ullapool, a large coastal village in the Scottish Highlands. He teaches children from ages 9-12. He was born in a fishing community north of Ullapool, where his father runs a marine fuels supply business and his mother is a community nurse. He has a degree in English from Glasgow University and retrained as a teacher after several years working as a web content author for a large leisure group.

Jack's experience as a web developer means that he is confident in all aspects of digital technology. He passionately believes that the effective use of digital technologies, blended with face to face teaching, can enhance the learning experience for children. He is particularly interested in using the iLearn system for project-based teaching, where students work together across subject areas on a challenging topic.

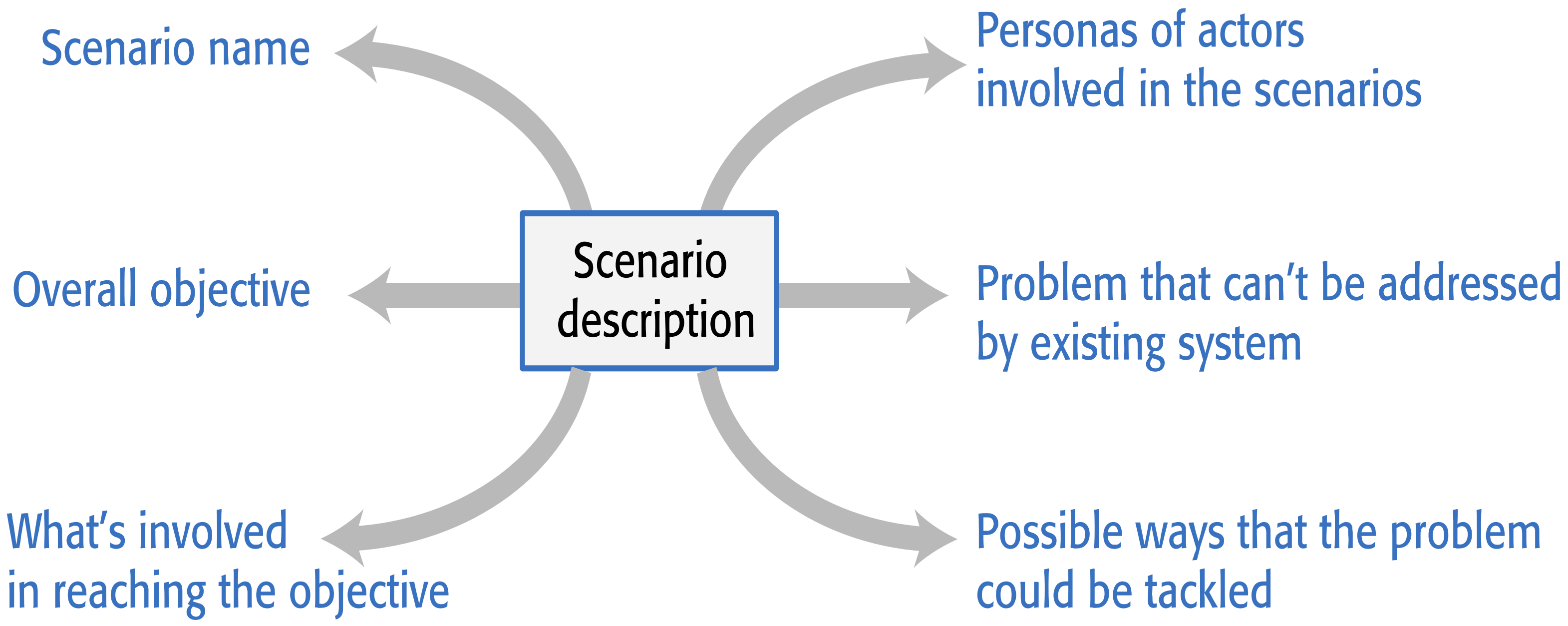
Scenarios and User Stories

- People find it easier to relate to real-life examples than abstract descriptions. They are not good at telling you the system requirements. However, they may be able to describe how they handle particular situations or imagine things that they might do in a new way of working.
- Stories and scenarios are ways of capturing this kind of information.

Scenario

- A scenario is a narrative that describes how a user, or a group of users, might use your system.
- There is no need to include everything in a scenario – the scenario isn’t a system specification.
- It is simply a description of a situation where a user is using your product’s features to do something that they want to do.
- Scenario descriptions may vary in length from two to three paragraphs up to a page of text.

Scenarios Description


Scenario description
Scenario name
Overall objective
What's involved in reaching the objective
Personas of actors involved in the scenarios
Problem that can't be addressed by existing system
Possible ways that the problem could be tackled

Scenarios (Example)

Fishing in Ullapool
Jack is a primary school teacher in Ullapool, teaching P6 pupils. He has decided that a class project should be focused around the fishing industry in the area, looking at the history, development and economic impact of fishing.

As part of this, students are asked to gather and share reminiscences from relatives, use newspaper archives and collect old photographs related to fishing and fishing communities in the area. Pupils use an iLearn wiki to gather together fishing stories and SCRAN (a history archive site) to access newspaper archives and photographs. However, Jack also needs a photo-sharing site as he wants students to take and comment on each others' photos and to upload scans of old photographs that they may have in their families. He needs to be able to moderate posts with photos before they are shared, because pre-teen children can't understand copyright and privacy issues.

Jack sends an email to a primary school teachers' group to see if anyone can recommend an appropriate system. Two teachers reply and both suggest that he uses KidsTakePics, a photo-sharing site that allows teachers to check and moderate content. As KidsTakePics is not integrated with the iLearn authentication service, he sets up a teacher and a class account with KidsTakePics.

He uses the the iLearn setup service to add KidsTakePics to the services seen by the students in his class so that, when they log in, they can immediately use the system to upload photos from their phones and class computers.

Scenarios (Elements of the Example)

- ✧A brief statement of the overall objective.
  - In Jack’s scenario, this is to support a class project on the fishing industry.
- ✧References to the personas involved (Jack) so that you can get information about the capabilities and motivation of that user.
- ✧Information about what is involved in doing the activity. For example, in Jack’s scenario this involves gathering reminiscences from relatives, accessing newspaper archives, etc.
- ✧An explanation of problems that can’t be readily addressed using the existing system.
  - Young children don’t understand issues such as copyright and privacy, so photo sharing requires a site that a teacher can moderate to make sure that published images are legal and acceptable.
- ✧A description of one way that the identified problem might be addressed.
  - In Jack’s scenario, the preferred approach is to use an external tool designed for school students.

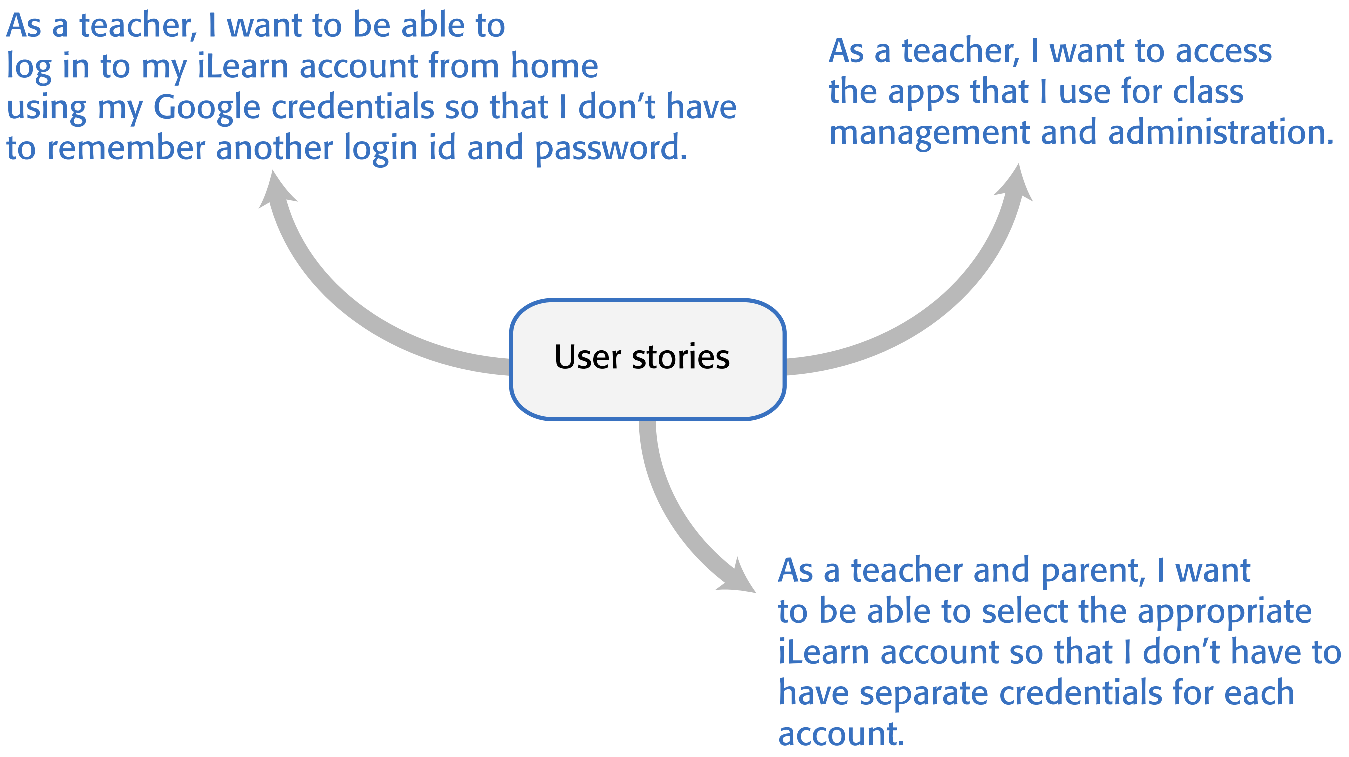
User Stories

- ✧Scenarios are high-level stories of system use. They should describe a sequence of interactions with the system but should not include details of these interactions.
- ✧User stories are finer-grain narratives that set out in a more detailed and structured way a single thing that a user wants from a software system.
  - *As an author, I need a way to organize the book that I'm writing into chapters and sections.*
- ✧This story reflects what has become the standard format of a user story:
  - ***As a*** *<role>, I <want | need>* ***to*** *<do something>*
    - *As a teacher, I want to tell all members of my group when new information is available*

User Stories

- ✧ An important use of user stories is in planning.
  - Many users of the Scrum method represent the product backlog as a set of user stories.
- ✧ User stories should focus on a clearly defined system feature or aspect of a feature that can be implemented within a single sprint.
- ✧ If the story is about a more complex feature that might take several sprints to implement, then it is called an epic.
  - As a system manager, I need a way to backup the system and restore either individual applications, files, directories or the whole system.

User Stories (Example)



As a teacher, I want to be able to log in to my iLearn account from home using my Google credentials so that I don't have to remember another login id and password.

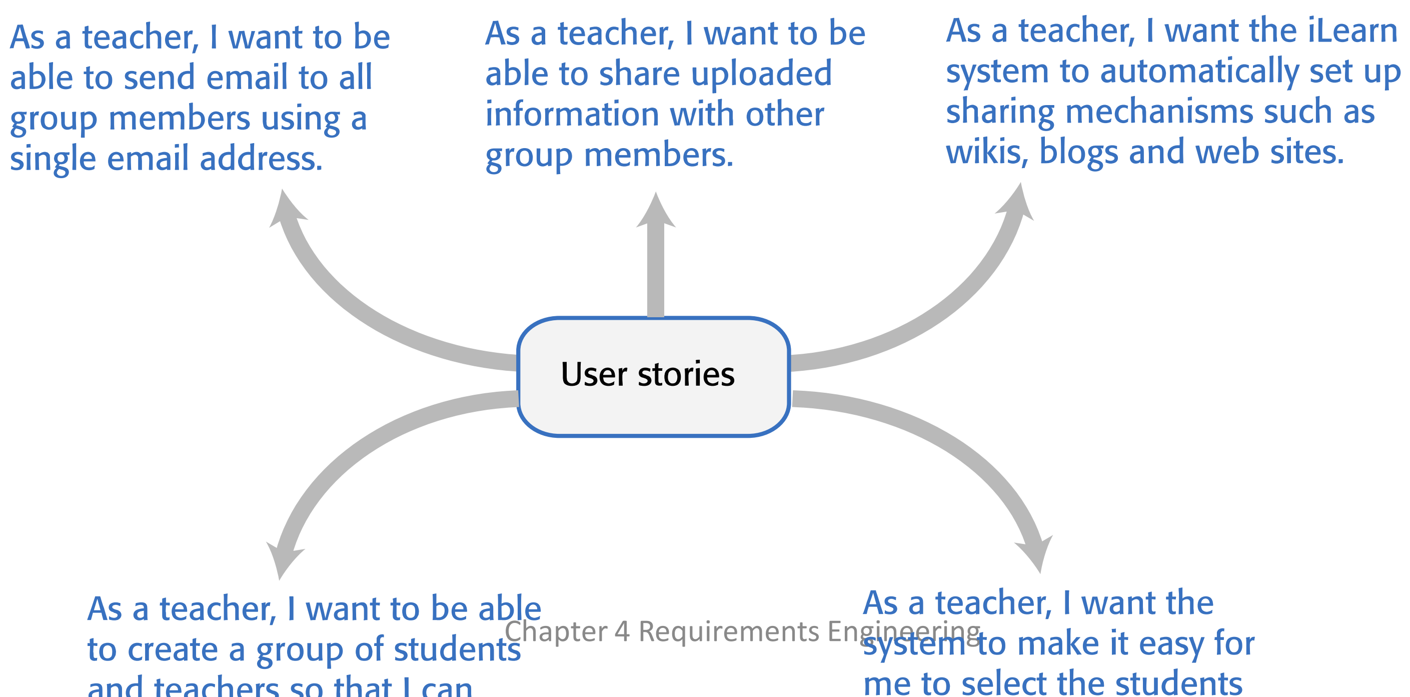
As a teacher, I want to access the apps that I use for class management and administration.

As a teacher and parent, I want to be able to select the appropriate iLearn account so that I don't have to have separate credentials for each account.

User Stories

✧ Stories can be used to describe features in your product that should be implemented.

✧ Each feature can have a set of associated stories that describe how that feature is used.



As a teacher, I want to be able to send email to all group members using a single email address.

As a teacher, I want to be able to share uploaded information with other group members.

As a teacher, I want the iLearn system to automatically set up sharing mechanisms such as wikis, blogs and web sites.

As a teacher, I want to be able to create a group of students and teachers so that I can

As a teacher, I want the system to make it easy for me to select the students

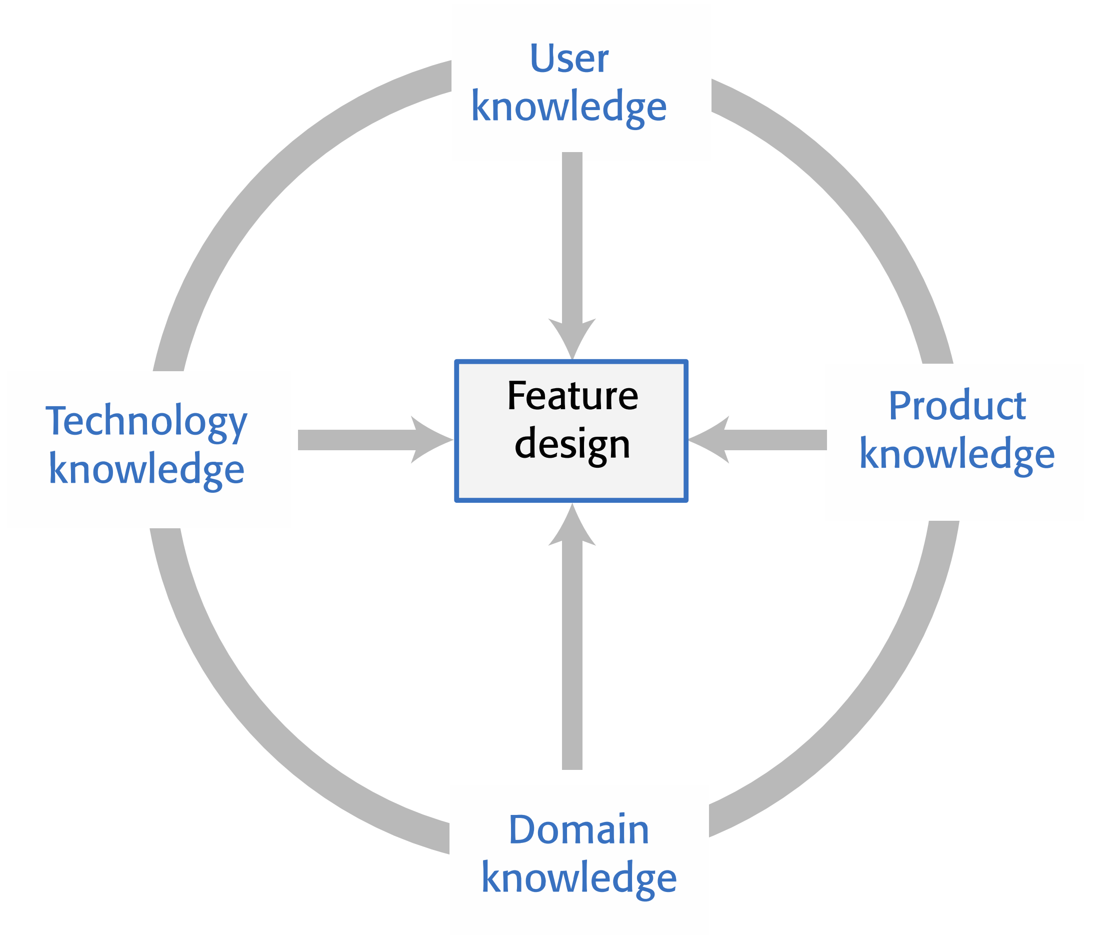
Scenarios vs User Stories

- Scenarios read more naturally because they describe what a user of a system is actually doing with that system. People often find it easier to relate to this specific information rather than the statement of wants or needs set out in a set of user stories.
- If you are interviewing real users or are checking a scenario with real users, they don't talk in the stylized way that is used in user stories. People relate better to the more natural narrative in scenarios.
- In some situation, scenarios can provide more context that user stories.

Features

- ✧A feature is a way of allowing users to access and use your product's functionality so the feature list defines the overall functionality of the system.
- ✧Features should be independent, coherent and relevant:
  - *Independence*
    Features should not depend on how other system features are implemented.
  - *Coherence*
    Features should be linked to a single item of functionality..
  - *Relevance*
    Features should reflect the way that users normally carry out some task.

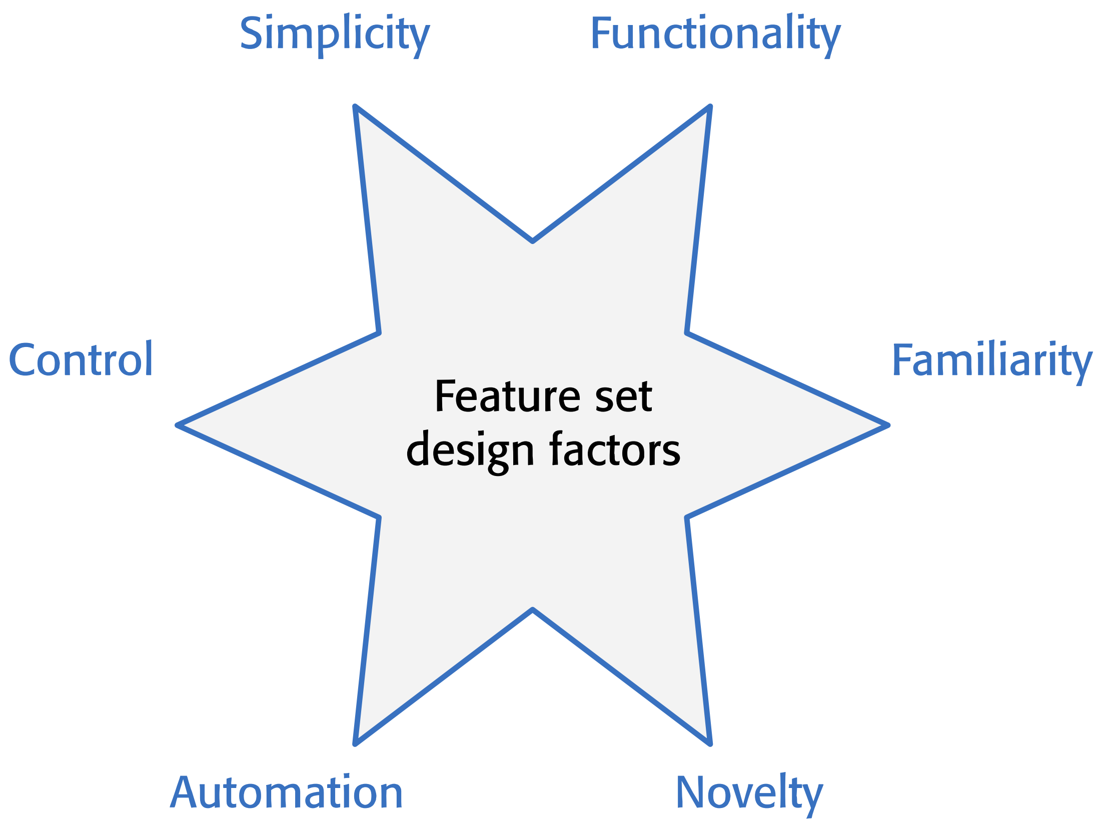
Knowledge Required for Feature Design



Knowledge Required for Feature Design

- ***User knowledge***
  You can use user scenarios and user stories to inform the team of what users want and how they might use it the software features.
- ***Product knowledge***
  You may have experience of existing products or decide to research what these products do as part of your development process. Sometimes, your features have to replicate existing features in these products because they provide fundamental functionality that is always required.
- ***Domain knowledge***
  This is knowledge of the domain or work area(e.g. finance, event booking) that your product aims to support. By understanding the domain, you can think of new innovative ways of helping users do what they want to do.
- ***Technology knowledge***
  New products often emerge to take advantage of technological developments since their competitors were launched. If you understand the latest technology, you can design features to make use of it.

Factors in Feature Design



Feature Tradeoff and Creep

- Simplicity and functionality
  - There should be balance between providing a simple, easy-to-use system and including enough functionality to attract users with a variety of needs.
- Familiarity and novelty
  - Users prefer that new software should support the familiar everyday tasks that are part of their work or life. There should be a balance between familiar features and new features.
- Automation and control
  - There need to be a balance between automation and manual tasks. The system should be tailored to the customers needs.
- Feature Creep
  - Feature creep occurs when new features are added in response to user requests without considering if these features are generally useful or can be implemented in some other way.

Features Identification

- ✧ Features can be identified directly from the product vision or from scenarios.
  - You should think about the features needed to support user actions, identified by active verbs, such as use and choose.
- ✧ Features in Jack's Scenario:
  - *A wiki for group writing.*
  - *Access to the SCRAN history archive. This is a shared national resource that provides access to historical newspaper and magazine articles for schools and universities.*
  - *Features to set up and access an email group.*
  - *A feature to integrate applications with the iLearn authentication service.*

Feature List

- The output of the feature identification process should be a list of features that you use for designing and implementing in the software.
- You can describe features using a standard input-action-output template by using scenarios and user stories.
  - Scenarios tell you how users work at the moment. They don’t show how they might change their way of working if they had the right software to support them.
  - Stories and scenarios are ‘tools for thinking’ and they help you gain an understanding of how your software might be used. You can identify a feature set from stories and scenarios.

Ambiguities- Drone system :

- ✧ The drone, a quad chopper, will be very useful in search and recovery operations, especially in remote areas or in extreme weather conditions. It will click high-resolution images. It will fly according to a path preset by a ground operator, but will be able to avoid obstacles on its own, returning to its original path whenever possible. The drone will also be able to identify various objects and match them to the target it is looking for.

Ambiguities- (Drone System)

1. **Extreme Weather Conditions**: The term "extreme" is subjective. Specify what environmental factors (e.g., wind speed, temperature, precipitation) are considered extreme to guide design and operational limits.
2. **High-Resolution Images**: Define the expected image resolution (e.g., in megapixels or specific dimensions like 4K) to ensure clarity in system requirements.
3. **Path Deviation**: Clarify the criteria or decision-making process the drone uses to deviate from its preset path and the conditions under which it returns to the original route.
4. **Object Identification Accuracy**: Specify the required accuracy or confidence threshold (e.g., 90% precision/recall) to evaluate the drone's object recognition capabilities effectively.
5. **Target Definition**: Clearly define what qualifies as a "target" (e.g., lost persons, specific equipment) to ensure proper training and functioning of the identification system.

Ambiguities- Ticket-issuing system:

- An automated ticket machine sells rail tickets. Users select their destination and input a credit card and a personal identification number. The rail ticket is issued and their credit card account charged. When the user presses the start button, a menu display of potential destinations is activated, along with a message to the user to select a destination and the type of ticket required. Once a destination has been selected, the ticket price is displayed and customers are asked to input their credit card. Its validity is checked and the user is then asked to input their personal identifier (PIN). When the credit transaction has been validated, the ticket is issued.

Ambiguities- (Ticket-issuing system)

1. **Select destination and type of ticket :** It's unclear what types of tickets are available (e.g., one-way, round-trip, first-class, child/senior fare).
2. **Credit card validity is checked :** Unclear what checks are performed—expiration date, available balance, stolen card status?
3. **Personal identifier (PIN) :** It’s not specified whether the system supports all card types and their PIN standards (e.g., chip-and-PIN vs. magstripe).
4. **Credit transaction has been validated :** Unclear what constitutes a successful validation—authorization only, or also completion?
5. **Ticket is issued :** Ambiguous whether the ticket is printed physically, sent digitally, or both.