

Requirements

CS 346 Application
Development

Credits

- Some content adapted from: Sommerville. 2021. [Engineering Software Products: An Introduction to Modern Software Engineering](#). Pearson. ISBN 978-1292376356.

Where do you start?

Start with a product vision.

The product vision

The starting point for product development is a '**product vision**'.

- Product visions are simple statements that define the essence of the product to be developed.
- The product vision should answer three fundamental questions:
 - What is the product to be developed?
 - Who are the target customers and users?
 - Why should customers buy this product?
- You should be able to articulate it in a couple of sentences.

Many paths to a product vision

- ***Domain experience***
The product developers may work in a particular area and understand the software that they need. They may be frustrated by deficiencies in existing software.
- ***Product experience***
Users of existing software may see simpler and better ways of providing comparable functionality and propose a new system that implements this e.g., new features.
- ***Customer experience***
The software developers may have extensive discussions with prospective customers to understand the problems that they face, constraints, and attributes that they need.
- ***Prototyping and playing around***
Developers may have an idea for software and might be involved in developing it into a product. They may develop a prototype system as an experiment.

Moore's vision template

Write a paragraph and fill in the following template:

FOR (target customer)

WHO (statement of the need or opportunity)

The (PRODUCT NAME) is a (product category)

THAT (key benefit, compelling reason to buy)

UNLIKE (primary competitive alternative)

OUR PRODUCT (statement of primary differentiation)

Vision template example

Example for an enterprise sales product.

FOR a mid-sized company's marketing and sales departments
WHO need basic customer-tracking functionality,
THE Sales-Innovator is a Web-based service
THAT provides sales tracking, lead generation, and sales representative
support features to improve customer relationships.
UNLIKE other services or package software products,
OUR product provides very capable services at a moderate cost.

Developing requirements

Build on your product vision to define a compelling product.

Software products

- There are three factors that drive the design of software products
 - **Business and consumer needs that are not met** by current products.
 - **Dissatisfaction** with existing software products, business or personal.
 - **Changes in technology** that make completely new types of product possible.
- In the early stage of product development, you are trying to understand:
 - Who are your users, and what are their challenges.
 - What product features would be useful to them, and
 - What they like and dislike about the products that they use.

Understanding users

- It makes sense in any product development to spend time trying to understand the potential users and customers of your product.
- A range of techniques have been developed for understanding the ways that people work and use software.
 - These include user interviews, surveys, ethnography and task analysis.
 - Some of these techniques are expensive and unrealistic for small companies.
- Informal user analysis and discussions, which simply involve asking users about their work, the software that they use, and its strengths and weaknesses are inexpensive and very valuable.

Software features

We often describe software in terms of the problem that it solves, and the features that it offers towards solving those problems.

- A feature is a fragment of functionality such as a 'print' feature, a 'change background feature', a 'new document' feature and so on.
- Before you start programming a product, you should aim to create a list of features to be included in your product.
- The feature list should be your starting point for product design and development.

User-Centered Design

We'll use three core concepts to help us define our features.

- **Personas:** an abstract representation of a user of a system.
- **Scenarios:** a high-level description of how a system will be used.
- **User Stories:** brief, informal descriptions of software features told from the perspective of the end-user.

We can use each of these to help us decide what product to build, and what features will be most useful.

A way of representing users

Personas

inspire

Natural language descriptions of a user
interacting with a software product

Scenarios

are-developed-into

Stories

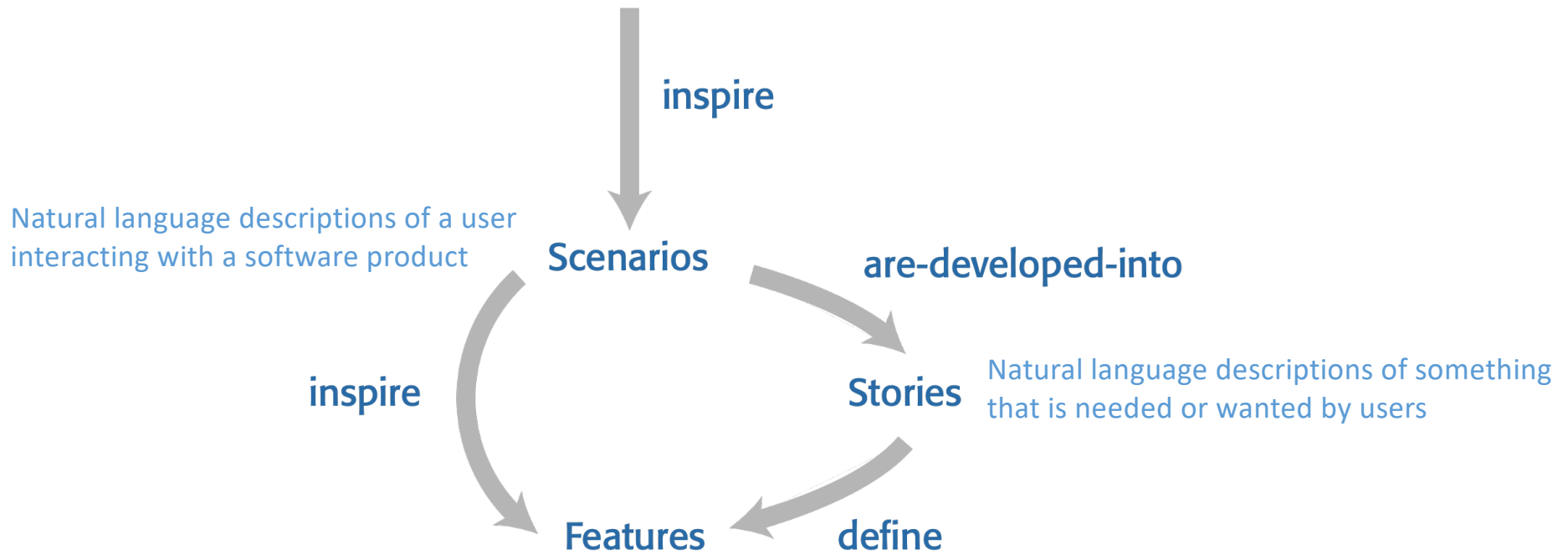
Natural language descriptions of something
that is needed or wanted by users

inspire

Features

define

How to figure out product functionality.



Personas

Characterizing your users.

Personas

- You need to understand your 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.
 - Create personas to reflect different aspects of your software.
 - Anticipate problems from the POV of each persona.

Persona: Jack

Concerned with educational software, in-class use, casual programming.

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 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.

Persona: Elena

Concerned with educational software, installation, maintenance, programming/customization.

Elena, age 28, is a senior IT technician in a large secondary school (high school) in Glasgow with over 2000 students. She was originally appointed as a junior technician but was promoted, in 2014, to a senior post responsible for all the school computers.

Although not involved directly in teaching, Elena is often called on to help in computer science classes. She is a competent Python programmer and is a 'power user' of digital technologies. She has a long-term career goal of becoming a technical expert in digital learning technologies and being involved in their development.

Jessica Coder

BIO

Jessica is a professional software developer. She graduated from UW with a BCS and a minor in CO. She is the tech lead for her current project at Shopify.

GOALS

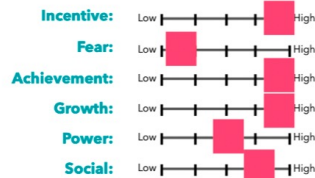
She is a packrat for saving technical information - code snippets, articles online.

FRUSTRATIONS

She finds it difficult to keep track of everything that she needs to work with in a day.

MOTIVATIONS

Anything to make her more efficient.



Age: 26

Occupation: Software Developer

Status: Single

Location: Toronto

Archetype: Developer

Personality: "Gadget person"; loves new tech; Android over iOS because she can tweak it.



QUOTES

"If I have to use a mouse, that just annoys me. I'm faster with a keyboard. VIM all the way".

"Information should be easy to access, and always accessible. Having everything searchable without breaking the flow would be amazing."

DOMAIN AWARENESS

Currently uses markdown notes and manually "greps" based on keyword.

TECH KNOWLEDGE

Expert in software, technology. Expects an advanced, polished solution.

Max Banker

BIO

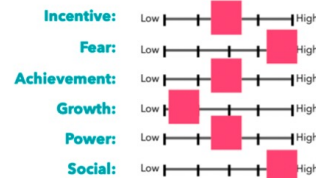
Max is a business student, with a political science minor. He has had multiple internships at banks and financial institutions. He's a capable technologist, but uses his computer for browsing the web and writing the occasional paper.

GOALS

Max has a lot of trouble keeping track of everything that he has to work on. Anything that would help him track course notes would be useful.

FRUSTRATIONS

Software is too complicated!



Age: 22

Occupation: Student

Status: Single

Location: Waterloo

Archetype: Student

Personality: Intense; serious; not particularly interested in tech.



QUOTES

"I'm too busy to learn some complex piece of software; it has to be easy to use."

"I hate my computer (Windows) so I'd rather use my phone."

DOMAIN AWARENESS

Not much. He scribbles everything in a paper notebook at the moment.

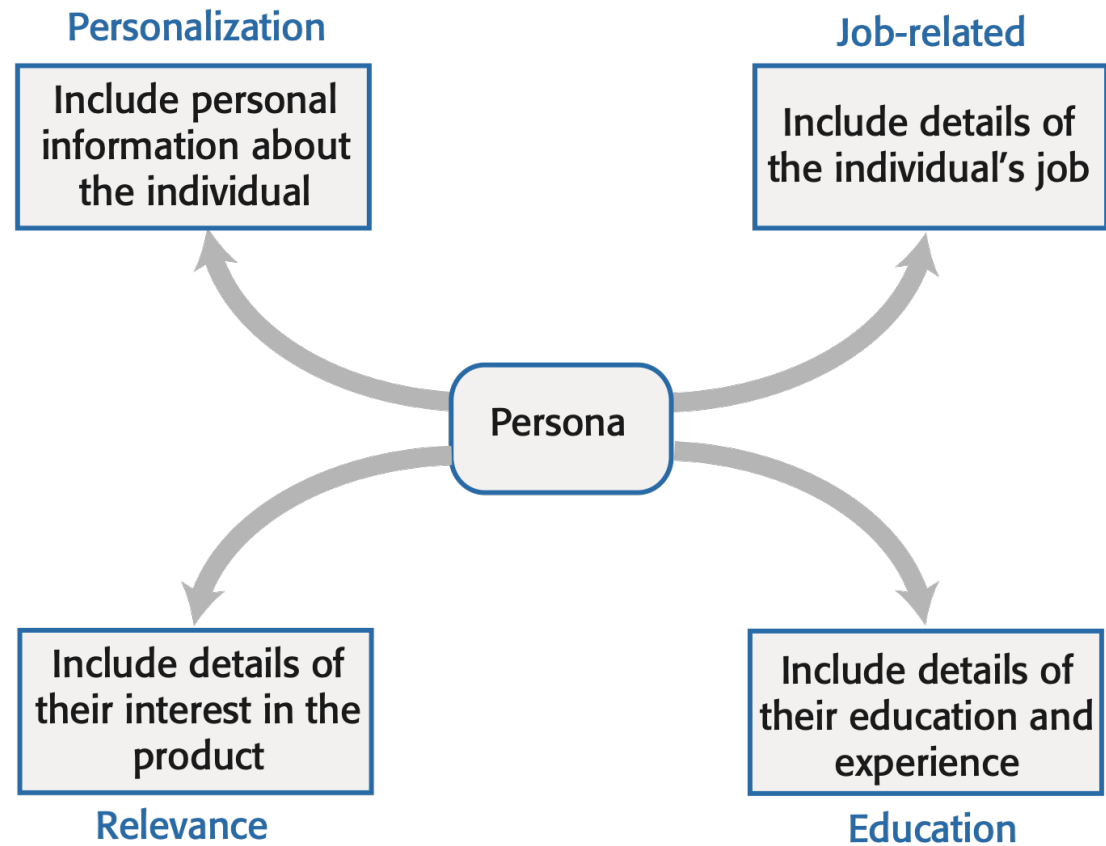
TECH KNOWLEDGE

Novice. He doesn't know much about computers.

Personas don't need to be narrative! Some people find summary cards like this more useful when brainstorming.

What makes a persona useful?

- 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.



Benefits of using personas

- The main benefit of personas is that they help you and other development team members empathize with potential users of the software.
- Personas help because they are a tool that allows developers to ‘step into the user’s shoes’.
 - Instead of thinking about what you would do in a particular situation, you can imagine how a persona would behave and react.
- Personas can help you check your ideas to make sure that you are not including product features that aren’t really needed.
- They help you to avoid making unwarranted assumptions, based on your own knowledge, and designing an over-complicated or irrelevant product.

Scenarios

A high-level description of how your system will be used.

Scenarios

- 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.

Scenario: Software in the classroom

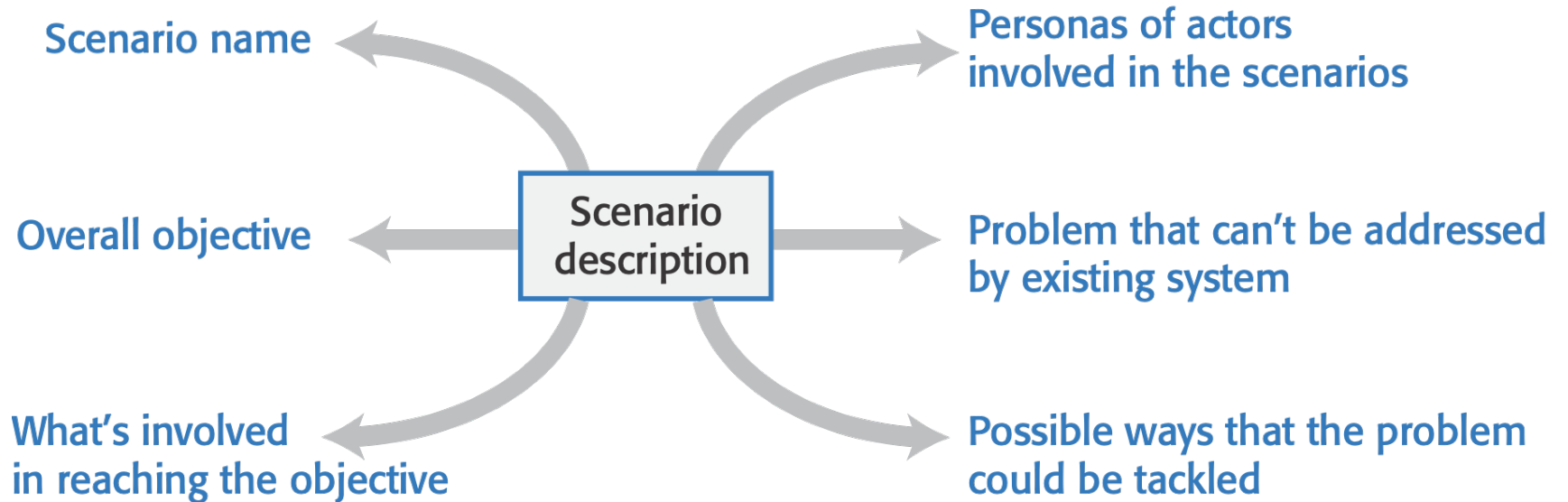
Jack is a primary school teacher in Ullapool, teaching P6 pupils. He has decided that a class project should focus on the local fishing industry, including its history, development and economic impact.

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 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 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.

Two teachers from a primary school teacher's group 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 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.

Structure of a scenario



Scenario elements

- **A brief statement** of the overall objective. e.g., supporting class projects.
- **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. e.g., in Jack's scenario this involves gathering reminiscences from relatives, accessing newspaper archives, etc.
- **An explanation of problems** that can't be addressed using the existing system.
 - Photos need to be vetted by teachers due to copyright concerns.
- **A description of a potential solution** that would be acceptable to stakeholders.
 - e.g., in Jack's scenario, the preferred approach is to design a tool for school students.

Writing scenarios

- Scenarios should always be written from the user's perspective and based on identified personas or real users.
- Your starting point for scenario writing should be your personas. You should normally try to imagine **several scenarios from each persona**.
 - Ensure that you have coverage of all potential user roles.
 - Personas may have different or even conflicting interests and involvement.
- Scenarios should be general and should not include implementation information (although describing implementation is often the easiest way to explain how something is done).
- NOTE: Scenarios are NOT specifications. You can't just implement it.

User involvement

- It is easy for anyone to read and understand scenarios, so it is possible to get users involved in their development.
- You should vet your scenarios with users that match your personas.
 - Develop an imaginary scenario based on our understanding of how the system might be used then ask users to explain what you have got wrong.
 - They might ask about things they did not understand and suggest how the scenario could be extended and made more realistic.
- Users are not good at writing scenarios.
 - They tend to focus on their individual lived-in experiences.
 - Don't ask them to write scenarios! (Better idea: talk to them and take notes).

User Stories

Descriptions of low-level system usage, from the perspective of a user.

Next: User Stories

- **Scenarios** are *high-level stories*.
- **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.
 - You define user stories by expanding and refining your scenarios.
- Standard format of a user story:
 - **As a** <role>, **I** <want | need> **to** <do something>
 - e.g., **As a** teacher, **I** want **to** tell all members of my group important information
- A variant of this standard format adds a justification for the action:
 - **As a** <role> **I** <want | need> **to** <do something> **so that** <reason>
 - e.g., **As a** teacher, **I** need **to** be able to report who is attending a class trip **so that** the school maintains the required health and safety records.

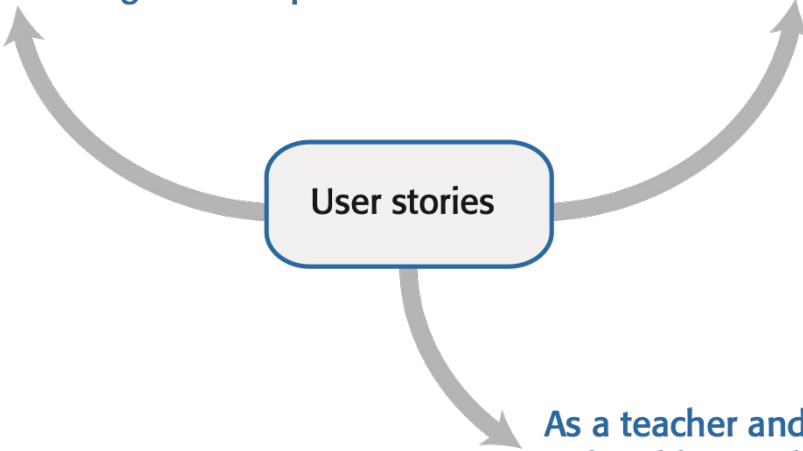
User stories in planning

- An important use of user stories is in planning.
- Many Scrum users 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.
 - During planning activities, you can expand into features (which are then assigned).
- If the story is about a more complex feature that might take several sprints to implement, then it is called an epic.
 - e.g., As a system manager, I need a way to backup the system and restore either individual applications, files, directories or the whole system.
 - There too much functionality associated with this user story. For implementation, it should be broken down into simpler stories with each story focusing on a single aspect of the backup system.

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.

User stories



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 from Emma's earlier scenario

Stories and scenarios

"If can express all the functionality described in a scenario as user stories, do you really need scenarios?"

- Scenarios are more natural and are helpful for the following reasons:
 - Scenarios read more naturally because they describe what a user is doing with that system. People often find it easier to relate to this style of narrative.
 - 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.
 - Scenarios often provide more context - information about what the user is trying to do and their normal ways of working. You can do this in user stories, but it means that they are no longer simple statements about the use of a system feature.

Feature Identification

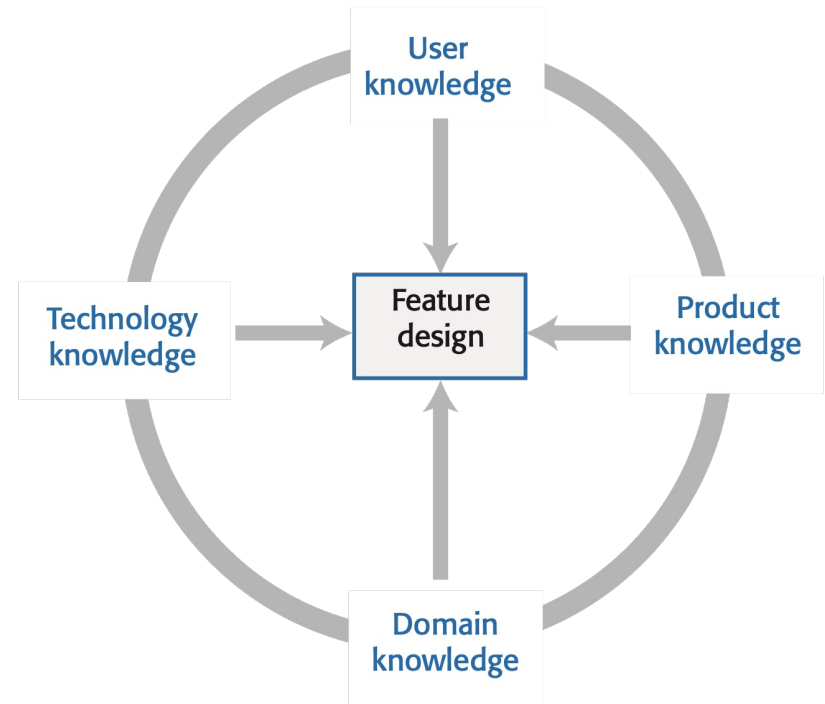
Implementation details derived from user stories.

Feature identification

- Your actual goal in the initial stages of product design should be to create a list of features that define your product. A 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, and they should never have side-effects.
 - **Relevance:** Features should reflect the way that users normally carry out some task. They should not provide obscure functionality.

Feature design considerations

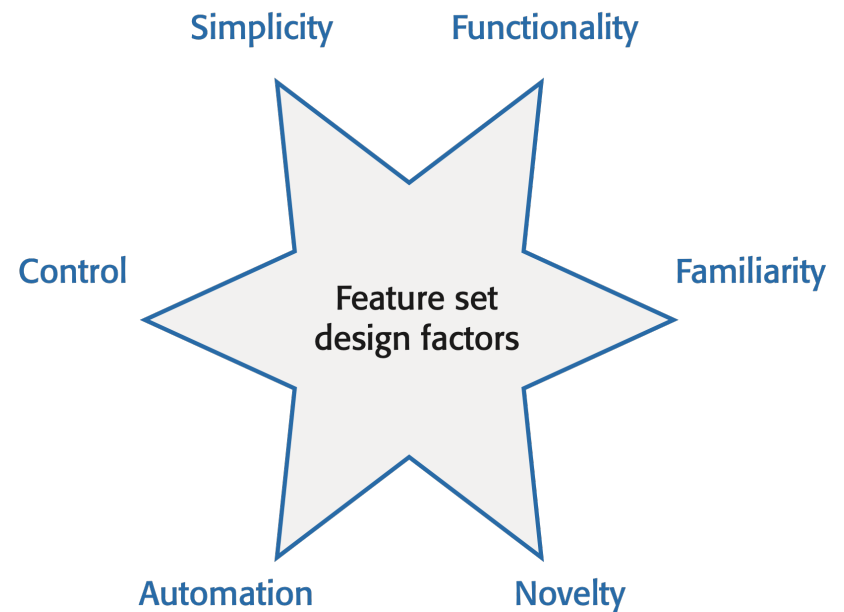
- **User knowledge:** User scenarios and user stories inform the team.
- **Product knowledge:** Investigate existing products as part of your development process e.g., work with existing features.
- **Domain knowledge:** Innovation requires understanding the problem domain e.g., finance, event booking.
- **Technology knowledge:** New products often emerge to take advantage of technological developments since their competitors launched.



Feature definition requires knowledge from each of these areas.

Feature trade-offs

- **Simplicity and functionality:** Balance a simple, easy-to-use system against adding more functionality to attract users.
- **Familiarity and novelty:** Users value familiarity, but they also want new features i.e., things your competitors may do. You need to balance familiarity with added complexity.
- **Automation and control:** Some users like automation, where the software does things for them. Others prefer to have control or provide more input into automation.



There are always design tradeoffs!

Feature creep

- Feature creep occurs when new features are added in response to user requests without considering whether these features are generally useful.
- Too many features make products hard to use and understand
- There are three reasons why feature creep occurs:
 - Product managers are reluctant to say 'no' when users ask for features.
 - Developers try to match features in competing products.
 - The product includes features to support both inexperienced and experienced users.

Does this feature really add anything new or is it simply an alternative way of doing something that is already supported?

Is this feature likely to be important to and used by most software users?

Feature questions

Can this feature be implemented by extending an existing feature rather than adding another feature to the system?

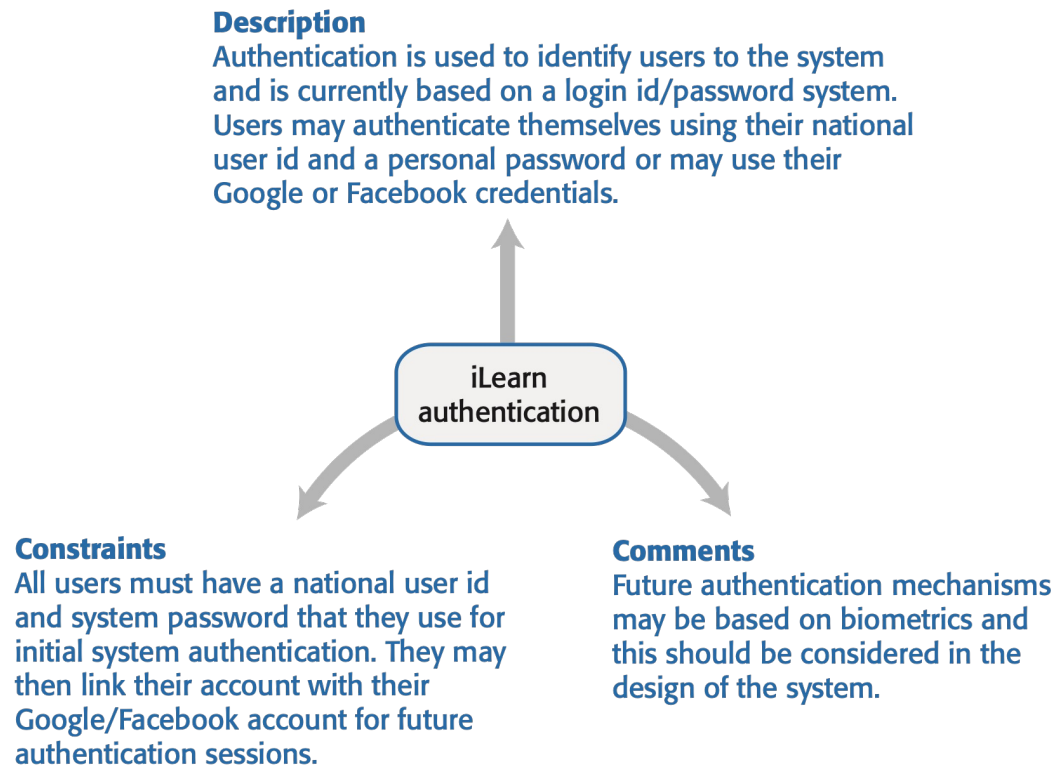
Does this feature provide general functionality or is it a very specific feature?

Things to consider when describing an issue.

Example: Features from Jack's scenario

We could extract the following requirements from Jack's scenario. Each of these would be logged in GitLab as an issue:

- 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.



Features should always contain a description of that feature. Constraints should be included when relevant i.e., restrictions, or dependencies on other features.

Innovation and feature identification

- Scenarios and user stories should always be your starting point for identifying product features.
 - 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.
- User research, on its own, rarely helps you innovate and invent new ways of working.
- You should also think creatively about alternative or additional features that help users to work more efficiently or to do things differently.

Summary 1

- A software product feature is a fragment of functionality that implements something that a user may need or want when using the product.
- The first stage of product development is to identify the list of product features in which you identify each feature and give a brief description of its functionality.
- Personas are 'imagined users' where you create a character portrait of a type of user that you think might use your product.
 - A persona description should 'paint a picture' of a typical product user. It should describe their educational background, technology experience and why they might want to use your product.
- A scenario is a narrative that describes a situation where a user is accessing product features to do something that they want to do.

Summary 2

- Scenarios should always be written from the user's perspective and should be based on identified personas or real users.
- User stories are finer-grain narratives that set out, in a structured way, something that a user wants from a software system.
 - User stories may be used as a way of extending and adding detail to a scenario or as part of the description of system features.
- The key influences in feature identification and design are user research, domain knowledge, product knowledge, and technology knowledge.
- You can identify features from scenarios and stories by highlighting user actions in these narratives and thinking about the features that you need to support these actions.

Reference

- Sommerville. 2021. [Engineering Software Products: An Introduction to Modern Software Engineering](#). Pearson. ISBN 978-1292376356.
- Shore & Warden. 2021. [The Art of Agile Development, 2nd Edition](#). O'Reilly. ISBN 978-1492080695.