Requirements

CS 346 Application Development

How do you begin?

Always start with a “big, ambitious idea”.

The product vision (aka *"idea"*)

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

- Product visions statements are simple statements that define the essence of the what you are developing.
- The product vision should answer three fundamental questions:
  - What is the product that we want to develop?
  - Who are the target customers / end-users?
  - Why should customers buy this product, instead of a competing product?

- You should be able to articulate it in a couple of sentences, or a short paragraph - also see elevator pitch.
- See brainstorming page on the course website

Examples: Vision Statement

“What’s for dinner” is a meal-planning application that takes the pain out of deciding what to cook for people who don’t have time or energy to meal plan! Use your phone to take a picture of your refrigerator, and our AI agent will compile a list of quick and easy recipes that you can make based on the ingredients you already have.

“Workout buddy” helps people who like to exercise keep track of their routines and week-by-week progress. You use it to build a workout plan consisting of sets, assign them to a schedule and then use it as part of your workouts. It helps motivate you by summarizing and charting your progress over time with fun graphics and animations.

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. As experts, they are in a great position to define a problem and viable solutions.

- ***Product experience***
  Users of existing software may see simpler and better ways of providing comparable functionality e.g., adding new features to functionality that you already use.

- ***Customer experience***
  Software developers may have discussions with prospective customers e.g., interviews 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 and test this prototype with users to gauge its effectiveness.

Format? Use Moore's vision template

The following template is a place to start:

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)

See Geoffrey Moore. Crossing the Chasm. 1991

Example: vision template

Example for an enterprise product that tracks sales and leads.

FOR a mid-sized company's marketing and sales departments
WHO need basic customer-tracking functionality,
THE Sales-Tracker 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.

Your vision statement should be the foundation of the rest of what you do; it should clearly express what your product does *and what* purpose *it serves.*

Even when you pivot (or drop features or add features) it should always be in service of this vision statement.

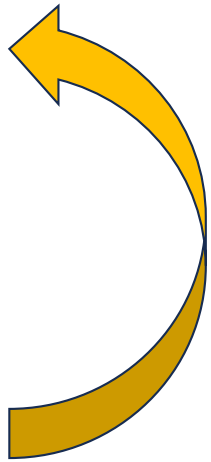
It's also intentionally broad in focus. We need to build on it before we start designing anything.

Developing requirements

Expand your product vision to define a compelling product.

Software products

- These 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 stages, you want to understand:
  - Who are your users, and what **needs are not being met**.
  - What product features **address their dissatisfaction**, and
  - What they like and dislike about the products that they use? What **improvements can we make** to existing solutions?

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.
- Let's formalize our approach.

User-Centered Design

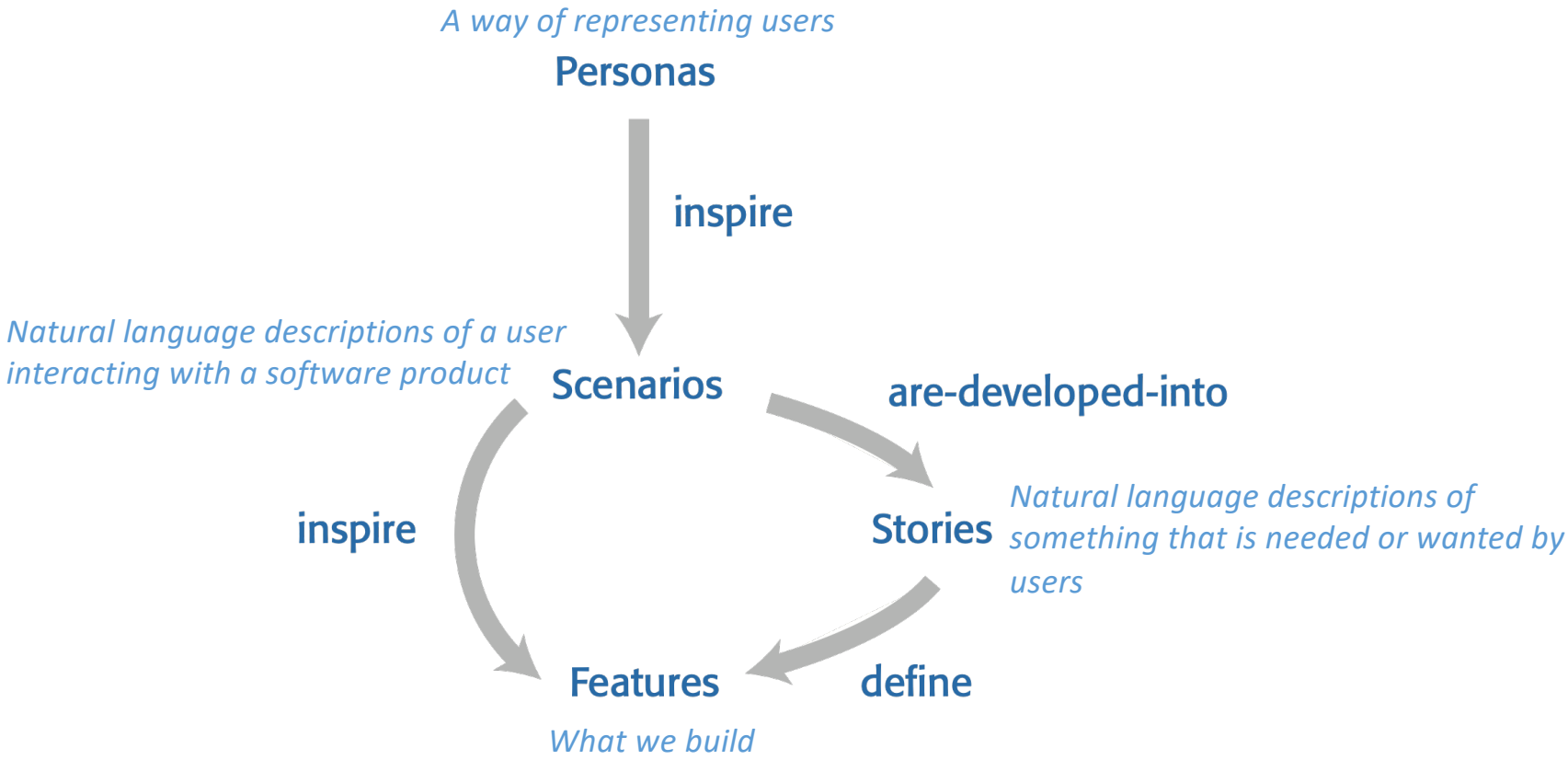
User-centered design focuses on identifying and meeting a user's needs.

- Process that considers user requirements throughout the product cycle.
- Based on work by Rob King (1977) and Don Norman (1986).

We'll use three core concepts from UCD 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.



The core of user-centred design is developing stories that describe how users interact with your product (Sommerville 2021).

UCD: 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 manages 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

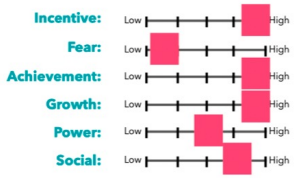
She is a packrat for saving technical information - code snippets, articles online.

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

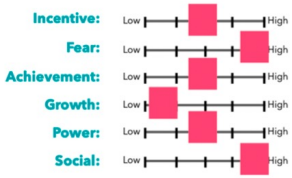
Max is a business student, with a political science minor. He has had multiple internships at banks and financial institutions. He's a capable technologist, but uses his computer for browsing the web and writing the occasional paper.

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. Choose whichever style you find helpful. Just make sure that it addresses your user's goals and motivations.

What makes a persona useful?

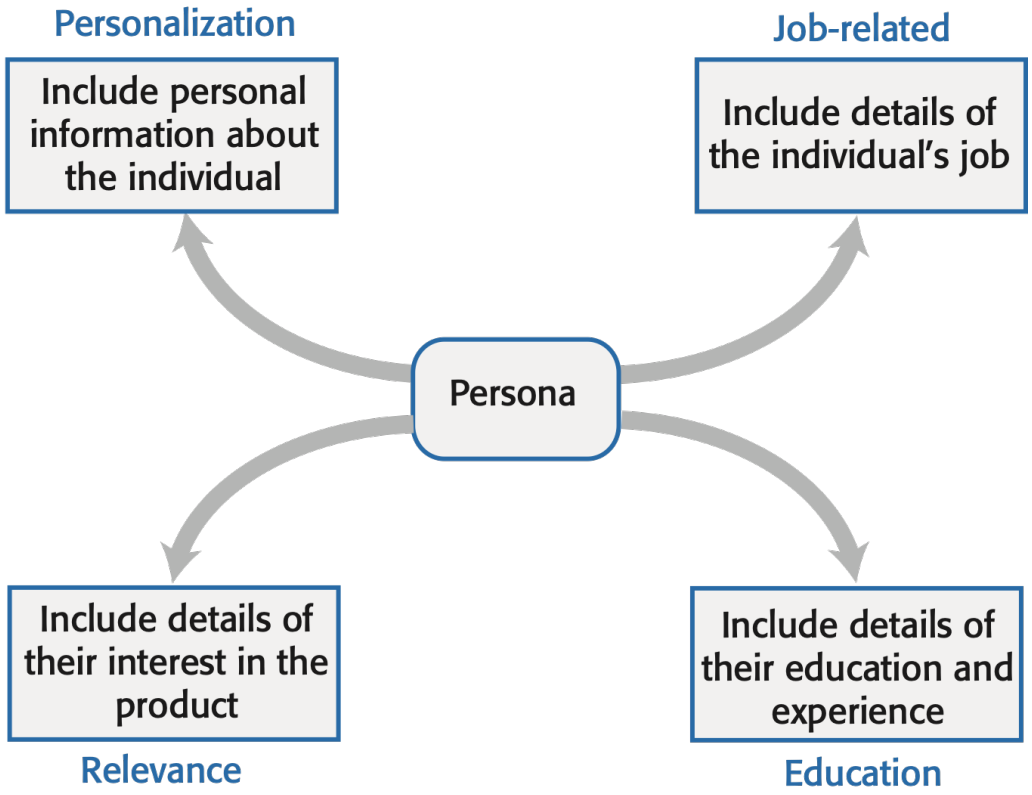
A persona should ‘paint a picture’ of a type of product user. They should be **relatively short** and easy-to-read but still **describe goals and motivation.**

A persona should help you assess whether a software feature is likely to be useful, understandable and usable by this type of user.

- Include expected background and why they might use your product.
- Personas should also include expected educational background and technical skills, as they relate to the problems you are examining.

It’s not uncommon to need multiple personas, reflecting different types of users.

There may be a primary or secondary personas that utilize different parts of your product.



What to include in a persona (Sommerville 2021).

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.
  - 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.

UCD: 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 complete system specification**.
  - It's 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: collecting photos for a class project

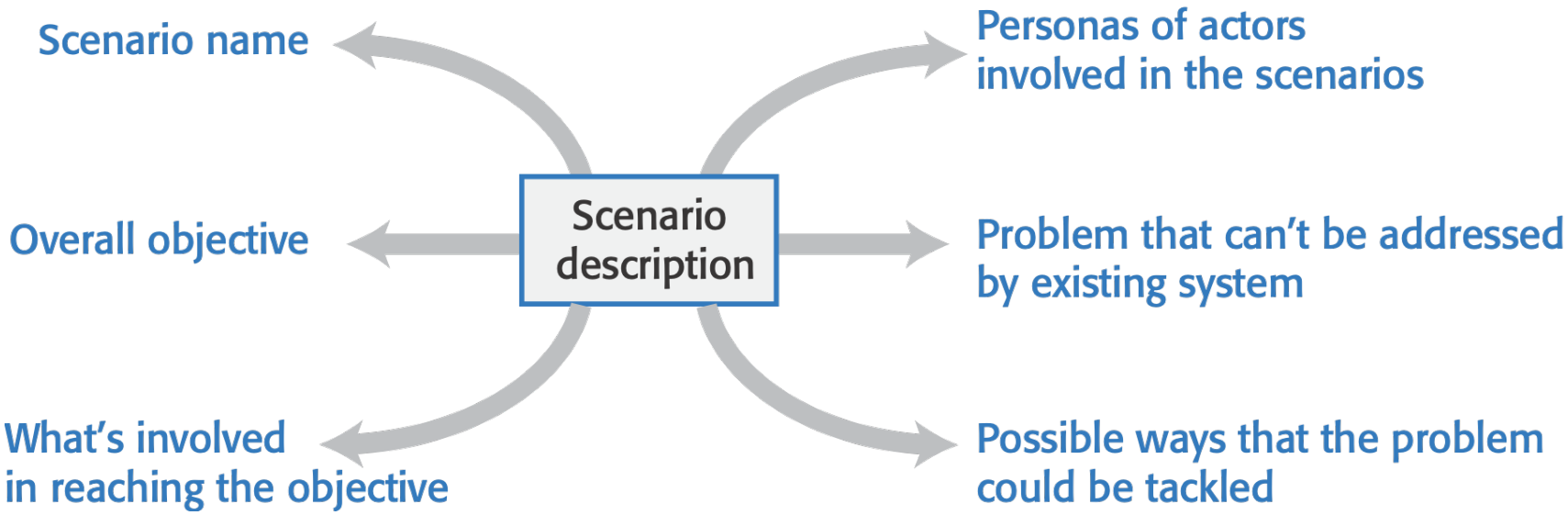
Jack is a primary school teacher in Ullapool, teaching P6 pupils. He has decided that a class project should focus on the local fishing industry, including its history, development and economic impact.

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. 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.*

Structure of a scenario



What to include in a scenario (Sommerville 2021).

Scenario elements

- **A brief statement** of the overall objective. e.g., supporting class projects.
- **References to the personas** involved (e.g., Jack, Elena) 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

- You should normally try to imagine **several scenarios from each persona**, to cover everything that they might do.
  - Every person should appear in at least one scenario.
  - You will often have multiple personas, interacting.
  - Personas may have different or even conflicting interests and involvement!
- Write scenarios from the perspective of the personas.
- Scenarios should be general and should not include implementation information from the developer's perspective
  - However, *describing* a possible implementation from the user's point of view may be the easiest way to explain how something should be done.

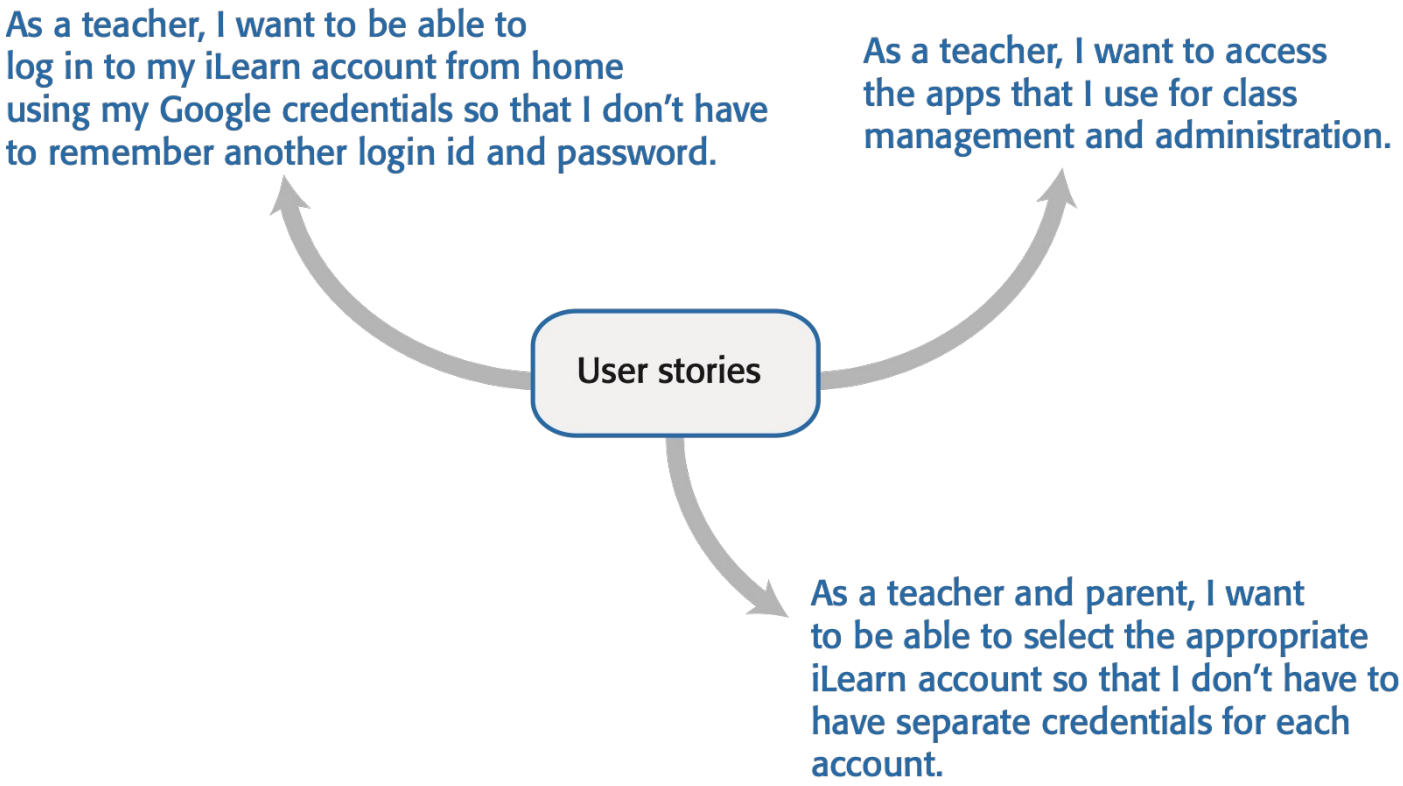
UCD: User Stories

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

Next: User Stories

- **Scenarios** are *high-level stories of system usage*. They describe context.
- **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; *they emerge from scenarios*.

- Standard format of a user story:
  - As a <role>, I <want | need> to <do something>
  - e.g., As a student, I need to be able to upload photos that I have found.
- 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 from Emma's earlier scenario

User stories in planning

- In Scrum, your **product backlog is a collection 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 aka issues.
  - You are always working towards addressing the features for a user story.
- 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!
  - It should be broken down into simpler stories with each story focusing on a single aspect of the backup system, which can then be implemented.

Stories and scenarios

“If can express all the functionality described in a scenario as user stories, do you really need scenarios?” (yes, you do)

Scenarios are helpful as a starting-point 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.
- 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

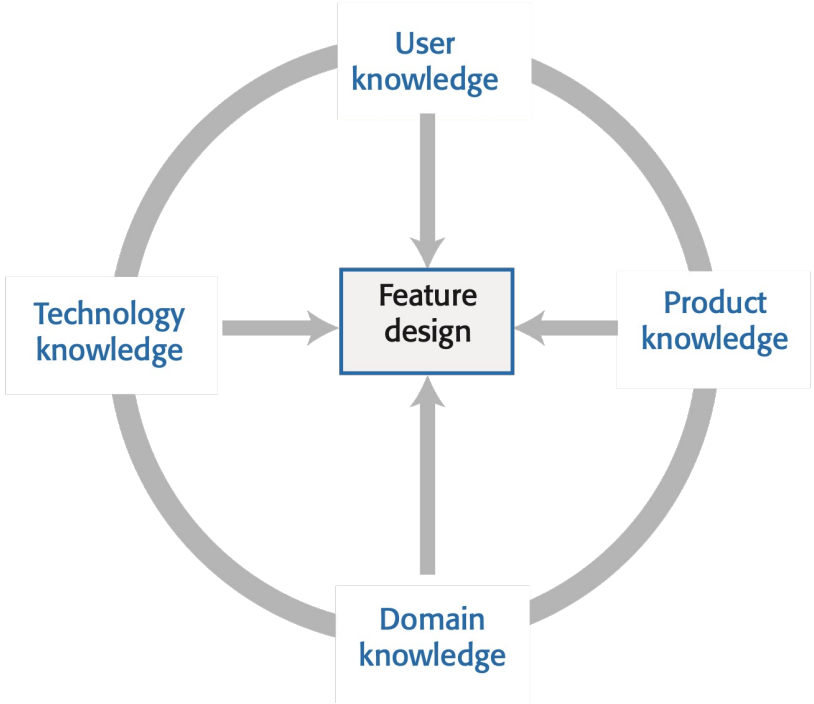
Implementation details derived from user stories.

Feature identification

- Your actual goal in the initial stages of product design should be to create a list of user stories, which contain the features that define your product.
  - Each user story contains 1 or more features.
  - You aim to complete the entire user story by delivering all of the features.
- Features should be independent, coherent and relevant:
  - **Independence**: Features should not depend on other system features outside of the user story. *Within* a user story they may be related.
  - **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.

Source of feature details?

- ***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.

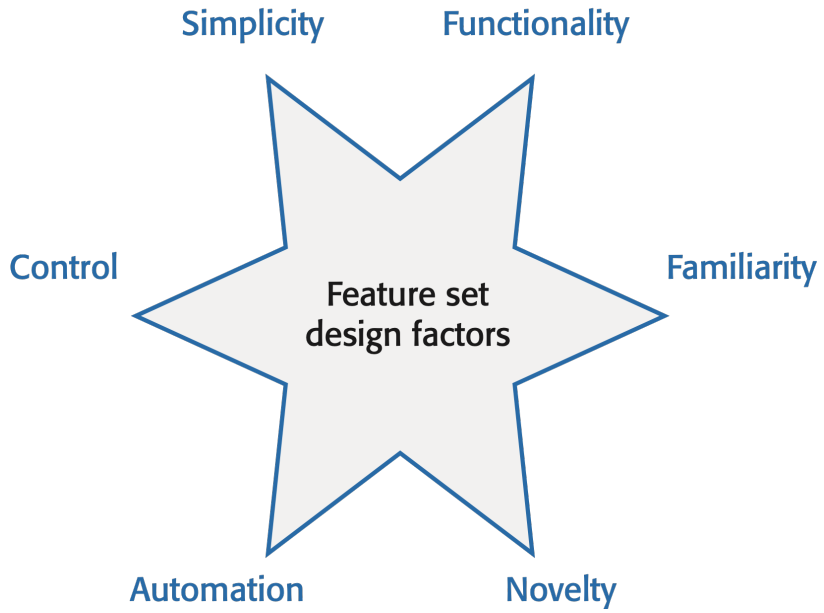


Feature definition requires knowledge from each of these areas.

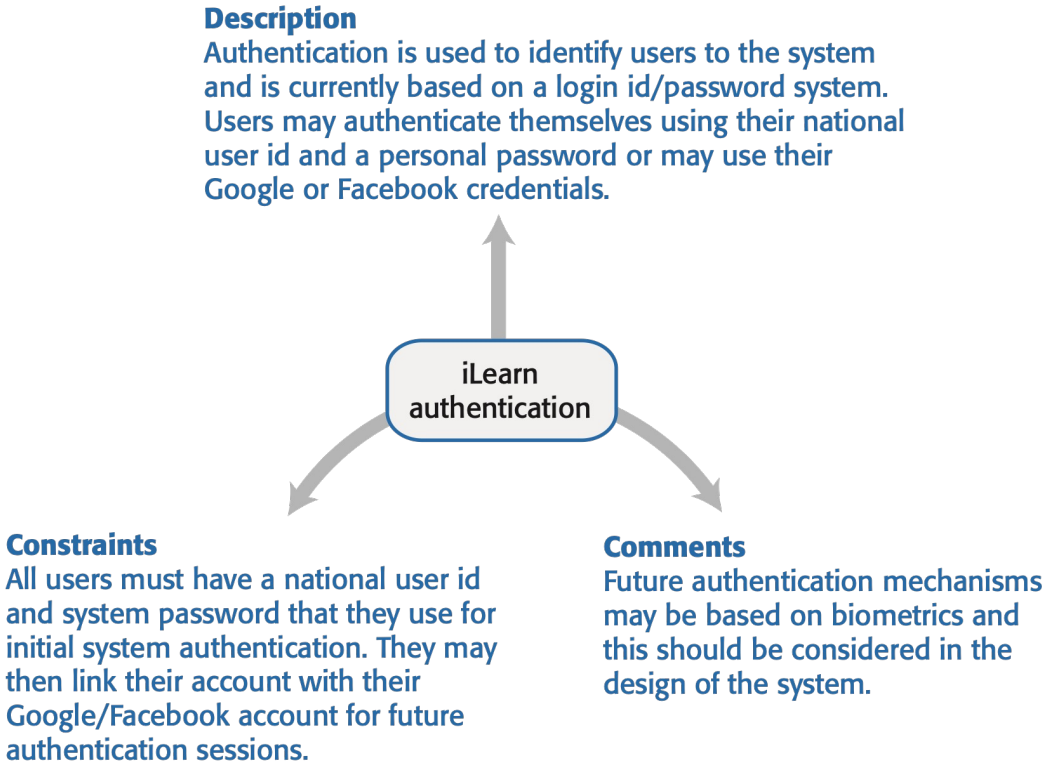
Warning: design involves trade-offs

You *probably* can't do everything that you have identified; watch for conflicting requirements.

- **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!



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 now. 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.

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.