
Software Documentation and Presentation



CEM YILMAZ
Technical University of Gabrovo
Faculty of
Electrical Engineering and Electronics

Notes made for the degree of
Software and Computer Engineering
January 22, 2026

Contents

1	Introduction	3
1.1	Module	3
1.2	Learning Objectives	3
2	Software Documentation	3
2.1	Technical Writing	3
2.2	Types of Documentation	4
2.3	Software Documentation Process	4
2.4	Markdown Formatting	5
2.5	Structured Writing	6
2.6	Graphics	7
2.7	Information Architecture	8
2.8	Cards Sorting	9
2.9	Quality	9
2.10	Style Guide	10
2.11	Content Management System (CMS)	11
3	Software Development Life Cycle (SDLC)	11
3.1	What is SDLC?	12
3.2	Phase 1: Planning	13
3.3	Phase 2: Requirements	13
3.4	Phase 3: Design and Prototyping	14
3.5	Phase 4: Software Development	15
3.6	Phase 5: Testing	15
3.7	Phase 6: Implementation	16
3.8	Phase 7: Operation and Maintenance	16
3.9	Waterfall Model	17
3.10	Incremental Model	18
3.11	Iterative Model	18
3.12	V Model	19
3.13	Spiral Model	20
3.14	Agile Model	21
4	Unified Modelling Language (UML)	22
4.1	What is UML	22
4.2	Class Diagrams	24
4.3	Composite Structure Diagrams	25
4.4	Package Diagrams	26
4.5	Component Diagrams	26
4.6	Deployment Diagrams	27
4.7	Object Diagrams	28
4.8	Profile Diagrams	29
4.9	Use Case Diagrams	30
4.10	Activity Diagrams	31
4.11	State Machine Diagrams	32
4.12	Sequence Diagrams	33
4.13	Communication Diagrams	34

4.14	Interaction Overview Diagrams	35
4.15	Timing Diagrams	36
5	Communication & Software Presentation	36
5.1	What is Communication	36
5.2	Three Principles of Communication	37
5.3	Three Levels of Understanding	38
5.4	Data and Information	38
5.5	Three Ingredients of a Presentation	39
5.6	How To Start a Presentation Effectively	40
5.7	Elements of an Effective Presentation	40
5.8	10 Tips for a Successful Presentation	42
5.9	8 Tips for Reducing Anxiety	43
5.10	Personal Appearance	43
5.11	10 Tips for Planning Successful Slides and Visualisation	44
5.12	15 Tips for Creating Better Slides	45
5.13	Layout	46
5.14	Teleconferencing and Videoconferencing	47
	Appendices	49
A	Style Guide Example	49
B	Planning & Design Example	65
C	SRS Example	75

1 Introduction

1.1 Module

This module is about showing, presenting and letting you plan a software product. You will be learning importance of information architecture in software documentation and how to effectively communicate with the end user or the developer. A documentation of a software may vary from a user guide to api documentation, metadata, video or even a specification and requirements of a document.

By the end of this module, you will be able to know the basics of writing software documentation and its strategies, how it can be presented, and various different media that can be used to describe the function of the software. In other words, you will be able to document, present and create graphics of a complex software product using tools such as Markdown, Powerpoint, UML and various other standards and technologies. You will gain the skills to show users and developers on how to set up and use the software or service, save time and energy, establish common rules and best practices, and assisting people solve problems by themselves.

1.2 Learning Objectives

The learning objectives for this module are:

- Learn what is required to start working on the software documentation for an app.
- Try out tools and infrastructure that help you immediately get started writing your software documentation.
- Learn Markdown language.
- Understand and write specific standards for writing software documentation, varying from global standards to company-specific standards.
- Learn how to present your software to users or colleagues confidently and correctly using MS PowerPoint.
- Learn Unified Modeling Language (UML).
- Learn methods of identifying a good documentation structure.
- Learn how to assess and identify quality software documentation.
- Learn the software development lifecycle (SDLC).
- Learn various SLDC structures used in corporations and identify the differences between them.

2 Software Documentation

2.1 Technical Writing

Technical writing is a discipline of writing text. It is a special type of writing used when you need to explain to another person: what is the product or services you provide and what are the different features of this product or service. In general documentation you should cover:

- Install
- Set up and configure
- Use
- Maintain
- Dispose of

your product or service. You have to plan, prepare, create, classify, deliver and maintain information for a specific software or service.

2.2 Types of Documentation

There are two types of documentation: functional and task-oriented documentation.

Definition 2.1. Functional Documentation

Functional documentation is suitable for users who are just getting started with a new product. It explains what you see in front of you, helps you find your way through the product and is often used for physical products.

An example of functional documentation is the Intro to Keynote on Mac. In this example, it is a high level example of the interface, with a short description paired with screenshots. In functional documentation, users are likely to only view the document once only. Usually, the steps are: start with the first screen, describe what is seen. You should organise the content in the order it appears in the UI.

Definition 2.2. Task-Oriented Documentation

Task-oriented documentation is suitable for advanced usage of a product, and it does not explain what is in front of you. It instead focuses on how to achieve a certain result. It guides you through the steps required to achieve that result, helping you find your way.

An example of task-oriented documentation is the Add or delete slides in Keynote on Mac. Specifically, it particularly provides steps on how to achieve a certain result. Usually, the steps are: identifying the tasks needed to perform, sequence them, add supporting concepts and information, and finally add reference information that helps them later on.

2.3 Software Documentation Process

The process of software documentation is as follows:

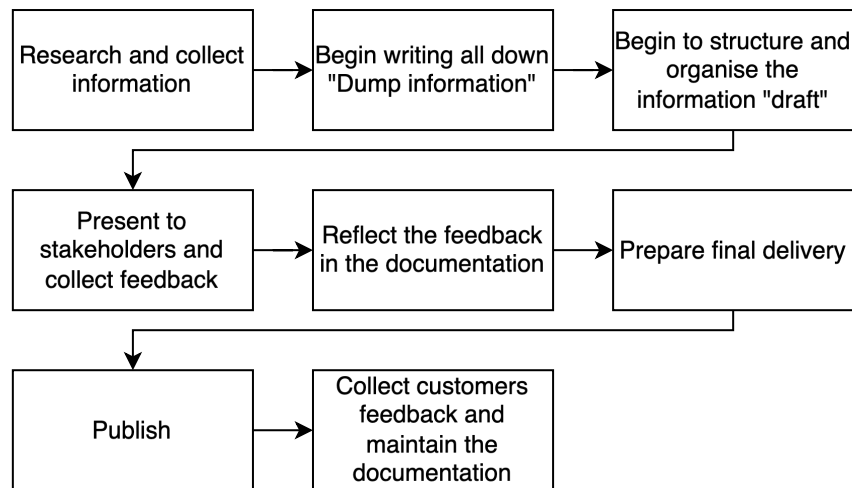


Figure 1: Software Documentation Writing Process

2.4 Markdown Formatting

Markdown is commonly used for basic documentation of software that is meant for developers to see. It is usually the default language that is used for READMEs of Git repositories. Additionally, GitHub has a "wiki" page dedicated to every repository, which can assist you in writing documentation for a specific software.

```

1  ** Bold **
2  * Italic *
3  ~~ Strikethrough ~~
4  ** Bold and _nested italic_ **
5  <ins>Underline</ins>
6
7  <sub>Subscript</sub>
8  <sup>Superscript</sup>
9
10 > Quote
11 ' Code Quote '
12
13 1. Enumerate
14 2. Enumerate
15 3. Enumerate
16 - Bullet point
17 - Bullet point
18 - Bullet point
19
20 > [!NOTE]
21 > This is note text.
22
23 > [!TIP]
24 > This is tip text.
25
26 > [!IMPORTANT]
27 > This is important text.

```

```

28
29 > [!WARNING]
30 > This is a warning text.
31
32 > [!CAUTION]
33 > This is caution text.
34
35 ![Alt image text](image url)
36
37 <a name="anchor-name"></a>
38 [Anchor text](#anchor-name)
39 [Anchor text](mailto:test@gmail.com)
40
41 Foornote reference[^1]
42 [^1]: Text located in the foornote
43
44 | col1 | col2 | col3 |
45 -----: | :----- | :-----:
46 | right-aligned text | left-aligned text | center text
47 |

```

Listing 1: GitHub MD Syntax

For examples of these markdowns and more detail, you can visit the official GitHub page on MD syntax. Another ready example of these elements used in a GitHub wiki can be found [here](#). We advise that you exercise by creating a GitHub public repository, and try out multiple functions such as creating pages.

2.5 Structured Writing

Definition 2.3. Structure Writing

Term coined by Robert E. Horn, his research identified over 200 common block types called information blocks which identifies common documentation types. They were then assembled into information types using information maps.

Structured writing is essential as it provides a better user experience (UX), can target different audiences, increase efficiency, organise content and ensure the completeness of the documentation. The most common information types are:

- Concept
- Procedure
- Process
- Principle
- Fact
- Structure

- Classification

An implementation of this theory in practice is the Darwin Information Typing Architecture (DITA) for the extensive markup language (XML). More information on DITA can be found [here](#). For example, the head of structured writing may have specific formats for specific tasks. These structure writing standards can vary from company to company and is called a style guide. When explaining a software concept (which answers the question what is...), it may chronologically always have:

- Title
- Summary (what is achieved? 1-2 sentences)
- Related (where could the answer also be?)
- Detailed Overview (video + technicality)
- Example (video)

Or for example, in a task (which answers the question how to...) you may have a format which is chronologically always:

- Title
- Summary
- Related (where could the answer also be?)
- Prerequisites
- Steps
- Result
- Example

Note that excessive amount of steps i.e. 10+, may indicate a problem, with little exceptions. Another possible topic is called "Reference". A reference is generally for advanced users that know the procedures, but require specific information e.g. specific parameters to do a function. They just care about the parameters, and not steps, as an example. It would appear like a list or a dictionary. The title or the heading of the documentation would typically be "About" or "What is" the software or service, and possibly listing its features.

2.6 Graphics

Graphics are a great way to present information to your readers without appearing boring. It is also up to 6000 times more efficient to perceive to end users compared to text. When writing documentation or presenting your software, it is important to add graphical elements that can aid the users in understanding the information. Some examples may include diagrams, UML, infographics, videos and other methods. There are several tools that let you achieve graphics, such as PowerPoint, Google Slides, Drawio, and even Canva. In this module, we will be specifically targeting PowerPoint and UML as different sections.

2.7 Information Architecture

Definition 2.4. Information Architecture

Information architecture is the art and science of organising large masses of information in a consistent and a logical manner.

In particular, for software documentation we need to ensure that the user can easily find what they're looking for. To do this, you will need clearly labelled, organised, and easily navigable documentation.

For effective information architecture, you need to pinpoint your audience. Specifically, not just as a label such as "hotel receptionists" or "software engineers", but rather the idea of who they are, their job, their education, their location, and how this can vary from person to person. As such, when designing software documentation, you need to brainstorm the type of people you are targeting and also do real research on the opinion of using your documentation for real feedback. You will notice large variations between used device, how they access information, and even the format they expect the documentation to be in. Prepare an interview with a list of questions, and even tasks of finding specific information, analyse it and adopt the changes in your documentation.

There are 6 ways of logically structuring information that is relevant for this course. They are:

- Categorical structure - presented topics are equally important, no hierarchy or sequence between categories, flexible order. E.g., A job listing describing skills e.g. teamwork, problem-solving, expertise etc.
- Evaluation structure - introduces a problem or decision, weights advantages and disadvantages, ends with a conclusion or recommendation. Used for advice and reviews e.g. pros and cons of changing jobs.
- Chronological structure - presented in a time order and focuses on how events unfold over time. Emphasises the process rather than the outcome. Used in reports, historical accounts. E.g., a narrative describing a historical event.
- Comparative structure - compares two or more viewpoints. Examines similarities and differences. Has criteria for comparison. Common in debates, arguments. E.g., a speech comparing political policies.
- Sequential structure - explains a process step by step with clear instructional language and fixed order of steps. Common in manuals, tutorials, etc. E.g. how to install software.
- Causal structure - explains causes and effects with no alternatives. Focuses on explaining why something happens. E.g. an article on the causes and effects of air pollution.

2.8 Cards Sorting

Definition 2.5. Card Sorting

A user research technique that allows you to create the information architecture from your content. There are two methods: open card sorting and closed card sorting. In open card sorting, you allow the users to group their cards in any way they want. Once they group them together, you ask them to label these groups. In closed card sorting, the users group their cards in pre-defined categories.

The benefits of the technique are:

- Decide on the structure of pages that lead to your content
- Label the content in a user oriented manner
- Group your content and/or products information in a way that makes sense to your target audience of users

The open card sorting technique allows you to compare the results of what a user expects of the documentation sequence to be like. In this technique, you let the user to do the sorting the cards which then you compare with your own implementation.

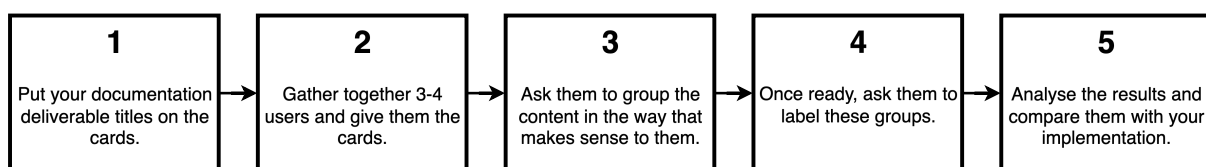


Figure 2: Open Cards Sorting Technique Process

You can also do this on a web-based software called Mural.

On the other hand, the closed card sorting technique has the categories completely pre-defined. The users group the titles together in the pre-defined groups. You can use this to measure how adequate your current navigation is from the customer perspective.

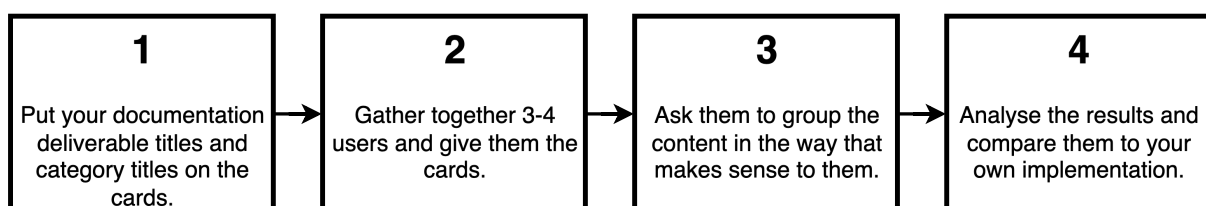


Figure 3: Closed Cards Sorting Technique Process

2.9 Quality

The definition of quality can vary from person to person, therefore, it is important to pin what a person means when they express their opinion on the software documentation. What do they mean by whether a software documentation is good, or what do they mean by when it is bad, and why. I.e., the definition can change from relative perspective to other documentation, or

whether it achieves the job it was intended to do. According to research by Jordan Stanchev, the characteristics of a high-quality software documentation are:

- Helpful - relevant, solves the problem, depicts the use cases.
- Comprehensive - covers all aspects, uses consistent terminology, satisfies the standard of the industry.
- Searchable - easy to find, easy search and navigation, SEO optimised, local search
- Visual - Incorporating short videos, using diagrams where possible, use of graphics, images, screenshots and annotated views.
- Usable - short and to the point, well organised content, providing the answer with minimum effort, links to detail when needed.
- Reliable - up-to-date, accurate, information from reliable source and validated.
- Grammatically Correct - easy to understand, no imperfections, no spelling mistakes, meaningful sentence structure and language.
- Translatable and Localisation Friendly - easy to localise, easy to translate to other languages
- Accessible - deployable by print or online, accessible at the point of need, to be in a shared and logical location.
- Uses Appropriate Tone - direct, simple, adheres to writing guidelines and rules for writing.

2.10 Style Guide

Style guides lets you a company to have their documentation, marketing and various other forms of writing to be consistent and fit with the branding. By ensuring consistency, order, and styling we are able to bring clear information to the target users. An example guideline can be found at University of Oxford's page. The style guide has general examples of good or bad, what you must do and what you mustn't do, etc. Even the style guides contains a logical order. The users of style guides are: usability experts, developers, owners, translators, content editors, user assistance developers, technical writers and more. A style guideline will help you achieve well organised content using:

- Conciseness - avoiding redundancies, using visual aids, link to information instead of copying text.
- Simplicity - simple language and grammar, short sentences, positive formulations, no long series of nouns, simple sentences.
- Precision - true information, consistent and correct terminology, navigation paths, product or component names.

It is also important to use precise verbs when addressing the steps or information in the guide. For example,

- MUST - an absolute requirement

- **MUST NOT** - an absolute prohibition
- **SHOULD** - a recommendation
- **SHOULD NOT** - not recommended
- **MAY** - optional

The key is to use precise keywords and in active, not passive voice. If appropriate, you may additionally address the user as "you". Ensure that terminology is explained clearly and consistently. In addition you can consider:

- Contractions - i.e. don't, can't, musn't
- Humour - i.e. avoid, it may be offensive in different cultures
- Gender - never assume the gender of the reader, use formulations such as his or her, them etc.
- Jargon or idioms - completely avoid

Always use tools such as Grammarly, spell-checkers, etc. to ensure precision and quality check. Even in a software's UI messages and text on screens e.g. errors, success messages should be compatible and possibly defined in the style guide.

2.11 Content Management System (CMS)

Definition 2.6. Content Management System

A software application that supports the creation and modification of digital content. It is typically used to support multiple users working in a collaborative environment. An example of this is Overleaf or Google Docs.

A CMS is used for development, review, storage, versioning, organising, structuring, publishing, error handling of a documentation. Some examples of CMS also include SDL, Ixiasoft, Schema ST4, and others. Generally a good CMS will let you output a single documentation in various formats such as HTML, PDF, ePub etc.

3 Software Development Life Cycle (SDLC)

3.1 What is SDLC?

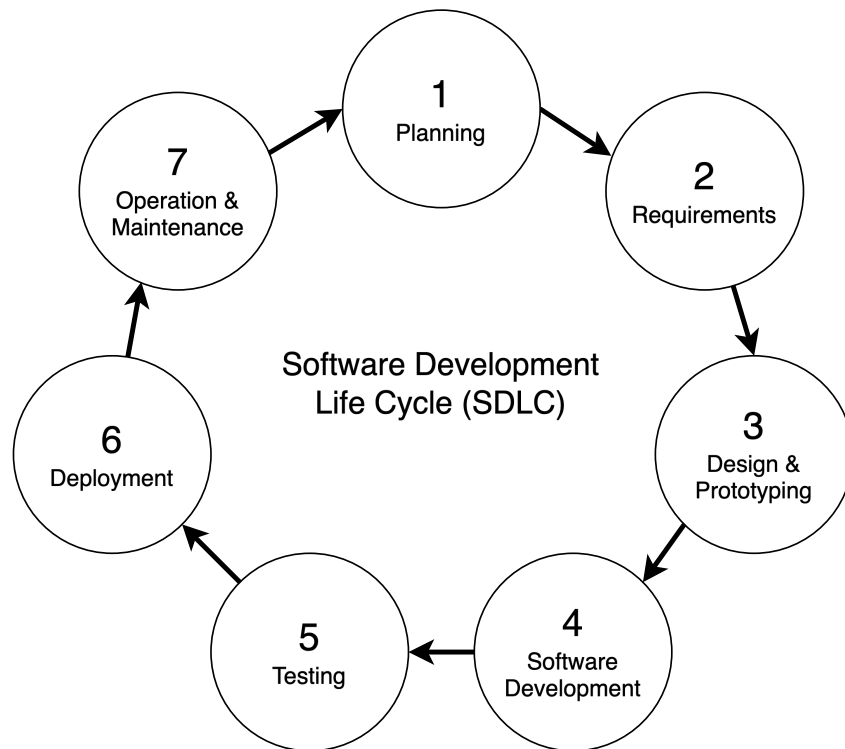


Figure 4: Software Development Life Cycle

The SDLC can be seen as a framework, process, model, guideline or even an approach to the development of software. It has a total of 7 stages, all of which will be explained in detail in this section. The stages are:

1. Planning
2. Requirement
3. Design and Prototyping
4. Software Development
5. Testing
6. Deployment
7. Operation and Management

The SDLC model exists to aid people in planning, to explain the process and to reach goals. The benefits of the SDLC model are:

- Tracking the development of the project and aiding its visibility.
- Establishing common vocabulary for each step of the process.
- Ensuring and improving quality of the software.
- Clarifying goals and responsibility of the software project.

- Increasing the speed of implementation
- Reducing the project risk
- Reduce project management costs
- Remove any unnecessary costs

3.2 Phase 1: Planning

In phase one, planning or analysis and planning, you analyse the alignment, resource allocation, scheduling and cost estimation of the software project. You would create a scope of work document (a type of documentation) where you would outline the project specifications, deliverables, timelines, agreements, risks and mitigations, etc. Some companies also include the prototypical design described through graphics using UML and other tools in the planning document to help plan the next section, requirements. An exemplar planning and design document can be found in Appendix B of this document.

3.3 Phase 2: Requirements

In phase two, requirements, you collect, research and analyse the requirements for the software project. You handle meetings with the business team, and the project analyst team which consist of the business analyst, project manager and the technical manager. You also analyse the use cases, such as how does the existing system work? Who will use the system? What input and output data are needed? Is it necessary to integrate with third-party interfaces (APIs) or tools? How will security and privacy be managed? What are the limitations?

At this stage, you would create the "Software requirements specification document", abbreviated as the SRS Document.

Definition 3.1. Software Requirements and Specification Document

A document that is created in phase 2 of SDLC. It contains the project's functional, non-functional and technical requirements.

In the SRS document, you would outline the functional requirements (what the system should perform, i.e., tasks) and non-functional requirements (how the system should perform, i.e. concepts).

Definition 3.2. Functional Requirements

Functional requirements of a software describe what the system should be performing. What the software application or product must be able to do. These are requirements usually requested by the end user. One could think of the are the "performed tasks" of the software.

Definition 3.3. Non-Functional Requirements

Non-functional requirements of a software describe how the system should be performing. Whether that is the accessibility, scalability, availability, maintainability or extensibility. One could think of these as the "architectural concepts" of the software.

Consider the exemplar below for an online banking system requirements:

Functional Requirements	Non-Functional Requirements
Users should be able to log in with their user-name and password.	The system should respond to user actions in less than 2 seconds.
Users should be able to check their account balance.	All transactions must be encrypted and comply with industry security standards.
Users should receive notifications after making a transaction.	The system should be able to handle 100 million users with minimal downtime.

Table 1: Functional and Non-Functional Requirements of Online Banking

The SRS document may contain the technical requirements.

Definition 3.4. Technical Requirements

The technical requirements of a software describe requirements such as the technical stack, packages, languages, databases, etc. The technical requirements may be included in the SRS or the planning document.

An exemplar SRS document can be found in Appendix B.

3.4 Phase 3: Design and Prototyping

In this phase, you would be converting the SRS, planning and design into a real prototype product. The idea is to create a minimum viable product (MVP) or sometimes a non-functioning prototype with barebones UI. The design document may additionally have two types of design: high-level design and low-level design

Definition 3.5. High-level Design

High-level design is a type of design documentation of a software found in phase 3 that concerns the entire system, such as describing the overall architecture, database design, dataflow diagrams, brief description of services, modules, relationships. It is produced by architects.

Definition 3.6. Low-level Design

Low-level design is a type of design documentation of a software found in phase 3 that concerns detailed descriptions of specific parts of the system. It would include the actual logic of each component and details of the specification of each module. E.g., classes. It is produced by the lead developers.

Once prototyping and designing is finished, in this phase you also collect prototype feedback for later stages and ensure improvement, and clear any miscommunications or unclear requirements with the business team. Note that the testing in this phase usually includes internal teams, and may not be completely thorough. The goal is to gain enough information for the final product.

3.5 Phase 4: Software Development

In phase 4, software development, you begin producing the final product. After gaining all the documents together, you will have a clear idea and a goal of what is required to be developed for the end user. At this stage, you may come across a developer guide that describes the best conventions during development. This may include things such as commenting your code, specific ways of naming functions, and generally how to maintain code. Note that document may not be present at all companies, as developers have general rules on naming conventions on things such as classes. These are general best practices. These conventions may even differ from language to language. An example for the HTML language can be found [here](#). The code will also likely be peer reviewed.

During the development phase, you may also come across the changelog. Changelogs typically can be seen through Git commit messages, or even in some cases, a full changelog that can be seen by end users. It describes the additions, removals or fixes per version change. If the code meets all the requirements, it then moves onto being tested by the quality assurance team.

3.6 Phase 5: Testing

In phase 5, testing, after the product has been developed, you begin with quality assurance testing. There are different types of testing that can happen:

- QA Testing - does the software product expected results in terms of quality i.e. ease of use, understanding?
- Functional Testing - do the functionalities of the software produce the results that are expected?
- Integration Testing - does the product perform the expected result and behaviour when calling APIs?
- Performance Testing - does the product perform specific functions, and integrations within specified times, number of users?
- Penetration Testing - how does the product perform on edge cases, rogue inputs or attempts to break the system?

If any tests fail, these are logged and informed to the development team. Successful passed tests are also logged. Note that these tests may either be automatic, manual or a mix. This phase will have subsequent releases.

Once the fixes are implemented to a satisfactory degree, a signature will be given from the testing team and the QA environment which will move onto the stage environment, where end

users will begin testing the product. This is called the UAT, user acceptance testing. If it is also satisfactory, signature once more will be collected to move onto the next phase. The final output is a detailed document that shows results of all types of tests.

3.7 Phase 6: Implementation

In phase 6, implementation, it is the phase at which you begin to launch your product. You enter the production deployment stage where all the code will be moved from a developer environment to a production environment. There are automated deployment tools such as Azure DevOps, Kubernetes with minimum manual intervention. Deployment activities are done by a DevOps team. In big companies, a request for change (RFC) is created. It is a formal document asking permissions for change, explaining step by step how the deployment is done. It additionally includes a rollback plan. Finally, a manager reviews this document, evaluates the risk and impact, and approves it.

3.8 Phase 7: Operation and Maintenance

In phase 7, operation and maintenance, the software enters its active lifecycle where it is being used by end users in the production environment. This phase focuses on ensuring the software continues to function properly and meets user needs.

Continuous monitoring of system performance, uptime, and user activity is essential during this phase. Technical support is provided to end users through help desks or support tickets, tracking and responding to bug reports and user feedback. Regular releases of updates, patches, and new versions are deployed, including security patches to address vulnerabilities, performance improvements, and documentation updates.

During this phase, Service Level Agreements (SLA) is typically established. The SLA contains:

- Establishing and maintaining SLAs that define expected service quality standards such as uptime percentage, response time, and resolution time
- Monitoring compliance with agreed-upon service levels (e.g., 99.9% uptime)
- Regular SLA reporting to stakeholders and clients

There are several types of maintenance that occur during this phase:

- Corrective Maintenance - Fixing bugs and defects that were not discovered during testing
- Adaptive Maintenance - Modifying the software to work with new operating systems, hardware, or third-party integrations
- Perfective Maintenance - Enhancing existing features or adding new functionality based on user requests
- Preventive Maintenance - Making changes to prevent future problems, such as code refactoring or performance optimization

This phase typically represents the longest period in the SDLC, as successful software products remain in operation for years, requiring ongoing attention and resources.

3.9 Waterfall Model

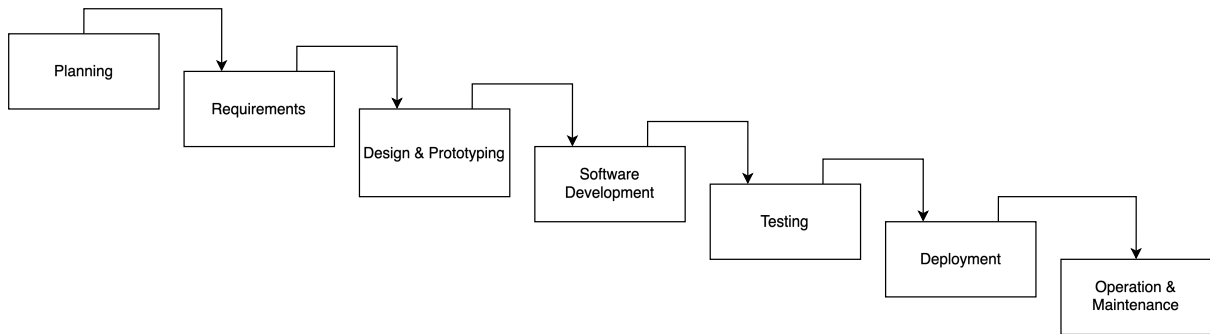


Figure 5: The Waterfall Model

The Waterfall Model is a sequential SDLC approach where each phase must be completed entirely before moving to the next phase. The process flows linearly through all seven phases: planning, requirements, design, development, testing, deployment, and maintenance, with no overlap between phases. Once a phase is completed, there is typically no going back to make changes. This model is almost never used in modern software development due to its inflexibility and inability to accommodate changing requirements. It may only be suitable for very small projects with extremely well-defined and stable requirements, or in highly regulated industries where extensive documentation and rigid processes are mandatory.

Advantages	Disadvantages
Simple and easy to understand and manage	No working software until late in the project lifecycle
Clear structure with well-defined phases and milestones	Extremely difficult and costly to accommodate changes
Easy to track progress and measure completion	High risk and uncertainty for complex projects
Works well for small projects with clear requirements	Assumes all requirements can be known upfront
Extensive documentation at each phase	Testing happens late, making bugs expensive to fix
Each phase has specific deliverables and review process	Not suitable for projects where requirements may evolve

Table 2: Advantages and Disadvantages of Waterfall Model

3.10 Incremental Model

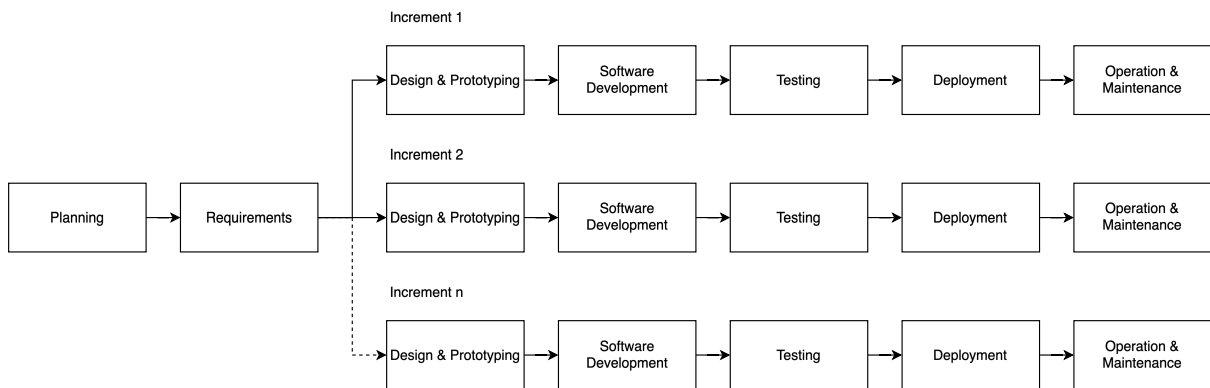


Figure 6: Incremental Model

The Incremental Model is an SDLC approach where the software is developed and delivered in small, manageable increments or modules. Each increment goes through all the SDLC phases (planning, requirements, design, development, testing, deployment, and maintenance), but only for a specific portion of the functionality. Instead of completing all phases for the entire system at once, you complete all phases for one increment at a time. Core features are typically developed in the first increment, with additional features added in subsequent increments, allowing users to start using basic functionality early before the complete system is finished.

Advantages	Disadvantages
Early delivery of working software allows users to provide feedback sooner	Requires good planning and design to define clear increments
Easier to test and debug smaller increments	May lead to integration issues between increments
Flexibility to change requirements between increments	Total cost may be higher due to repetitive phases
Lower initial delivery costs and risks	Requires more management and coordination effort
Easier to manage and prioritise critical features first	System architecture must be defined early and remain stable

Table 3: Advantages and Disadvantages of Incremental Model

3.11 Iterative Model

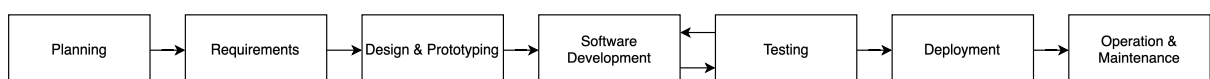


Figure 7: Iterative Model

The Iterative Model is an SDLC approach where the software is developed through repeated cycles or iterations. Each iteration goes through all the SDLC phases (planning, requirements, design, development, testing, deployment, and maintenance) and produces a working version of

the software. Unlike the Incremental Model where each increment adds new functionality, the Iterative Model refines and improves the entire system with each iteration based on feedback and testing results. The first iteration typically implements a basic version of the complete system, and subsequent iterations enhance, optimise, and add features to the existing system. This model is particularly useful when requirements are not fully understood at the beginning or are expected to evolve over time.

Advantages	Disadvantages
Early working prototype available for user feedback	Requires more resources and time due to repeated cycles
Easier to accommodate changing requirements	Can lead to scope creep if not properly managed
Risks are identified and resolved early in iterations	Requires strong management and clear objectives for each iteration
Continuous testing and refinement improves quality	May be difficult to define clear iteration boundaries
Parallel development can occur across different iterations	Architectural changes in later iterations can be costly
Better suited for large and complex projects	Requires highly skilled and experienced team members

Table 4: Advantages and Disadvantages of Iterative Model

3.12 V Model

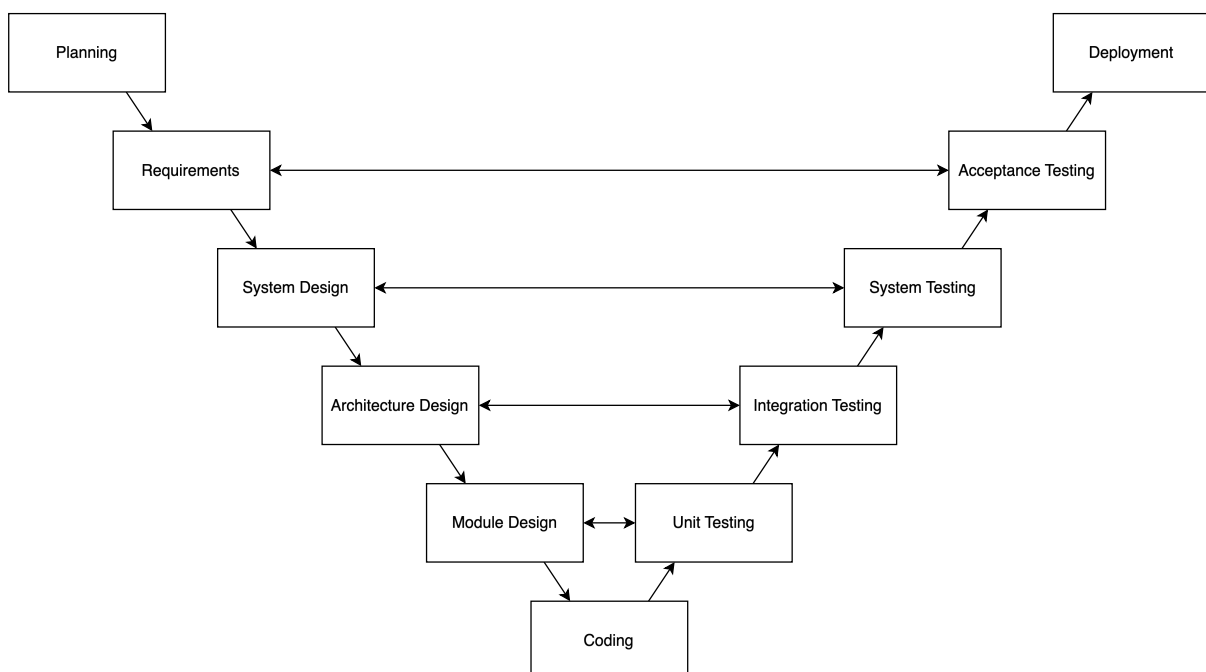


Figure 8: V Model

The V Model, also known as the Verification and Validation Model, is an extension of the Waterfall Model that emphasises testing at each phase of development. The model is shaped like the

letter "V" where the left side represents the development phases descending from requirements to implementation, and the right side represents the corresponding testing phases ascending from unit testing to acceptance testing. Each development phase on the left has a corresponding testing phase on the right: requirements correspond to acceptance testing, design corresponds to system testing, and detailed design corresponds to integration and unit testing. This model ensures that testing is planned in parallel with development, making it particularly suitable for projects where quality and reliability are critical, such as medical devices, aviation systems, or safety-critical applications.

Advantages	Disadvantages
High emphasis on testing and quality assurance	Very rigid and inflexible like the Waterfall Model
Testing is planned early alongside development phases	No working software until late in the development cycle
Clear and well-defined phases with specific deliverables	Difficult to accommodate changing requirements
Easy to track progress and defects at each stage	Not suitable for complex or long-term projects
Works well for small to medium projects with clear requirements	High risk for projects with uncertain or evolving requirements
Defects are found early due to parallel test planning	Expensive to implement changes once testing has begun

Table 5: Advantages and Disadvantages of V-Model

3.13 Spiral Model

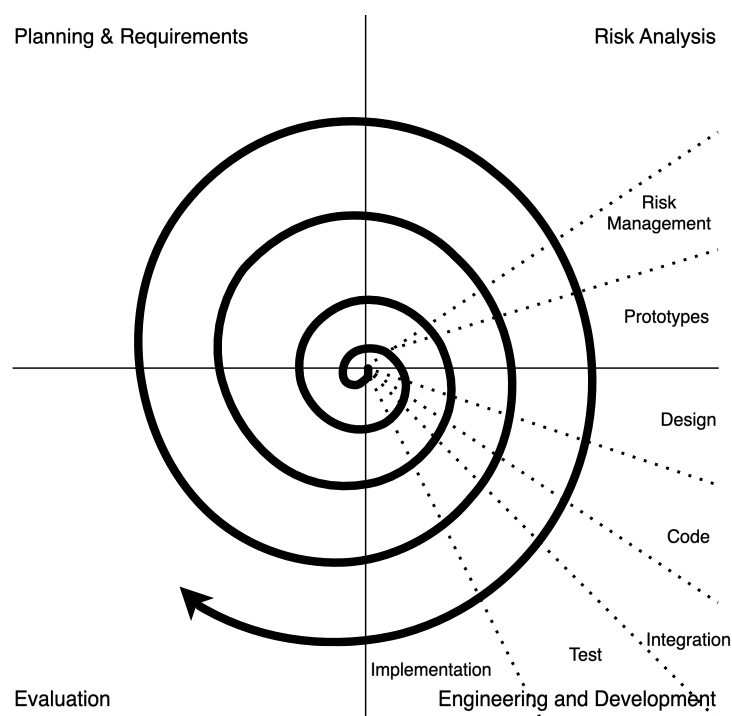


Figure 9: Spiral Model

The Spiral Model is an SDLC approach that combines elements of both iterative development and the Waterfall Model, with a strong emphasis on risk analysis. The model is represented as a spiral with multiple loops, where each loop represents a phase in the development process. Each spiral iteration consists of four main quadrants: planning and requirements gathering, risk analysis, engineering and development, and evaluation by stakeholders. The process starts from the center of the spiral and progresses outward with each iteration, allowing the project to evolve incrementally while continuously assessing and mitigating risks. This model is particularly suitable for large, complex, and high-risk projects where thorough risk assessment is critical, such as enterprise systems, mission-critical applications, or projects with significant uncertainty in requirements or technology.

Advantages	Disadvantages
Strong emphasis on risk analysis and mitigation	Can be expensive due to extensive risk analysis activities
Flexibility to accommodate changing requirements	Requires highly skilled risk analysis expertise
Early identification of potential risks and issues	Not suitable for small or low-risk projects
Continuous stakeholder involvement and feedback	Complex to manage and requires extensive documentation
Suitable for large and complex projects	Success heavily depends on risk assessment accuracy
Prototypes are developed early for validation	Can lead to scope creep if not properly controlled

Table 6: Advantages and Disadvantages of Spiral Model

3.14 Agile Model

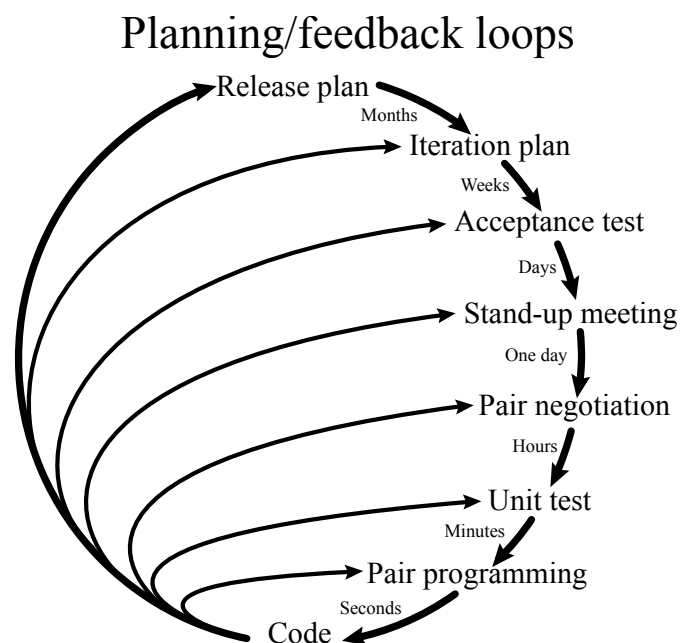


Figure 10: Extreme Programming Model

The Agile Model is a modern SDLC approach that emphasises flexibility, collaboration, and rapid delivery of working software through iterative and incremental development. The development process is divided into short time periods called sprints, typically lasting 1-4 weeks, where a potentially shippable product increment is delivered at the end of each sprint. Agile focuses on continuous collaboration between self-organising cross-functional teams and stakeholders, with regular feedback and adaptation to changing requirements. The model prioritises working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a fixed plan. Popular Agile frameworks include Scrum, Kanban, and Extreme Programming (XP). Agile is currently the most widely used SDLC model in modern software development, particularly for web applications, mobile apps, and projects where requirements are expected to evolve.

Advantages	Disadvantages
Rapid delivery of working software in short iterations	Requires significant customer involvement and commitment
High flexibility to accommodate changing requirements	Can be difficult to predict timelines and costs accurately
Continuous customer feedback and collaboration	Less emphasis on documentation can cause knowledge gaps
Early and frequent detection of issues and defects	Requires highly skilled and experienced team members
Improved team morale through self-organisation	Not suitable for projects with fixed scope and budget
Better quality through continuous testing and integration	Can lead to scope creep without proper discipline

Table 7: Advantages and Disadvantages of Agile Model

4 Unified Modelling Language (UML)

4.1 What is UML

Unified Modelling Language is a standard for creating schemas, diagrams, and figures in a way that explains the flow of a software, service or a product. UML was developed by software engineers Grady Booch, Ivar Jacobson and James Rumbaugh during 1994 and 1995. Today it is maintained by the Object Management Group (OMG) and the IEEE. Latest documentation of the UML version can be found [here](#). For drawing diagrams in a software, we recommend using drawio.

In this section, we will be covering in what specific scenarios UML is used and how. We will not be teaching the symbols of all UML standards as the symbol list is massive. To find the symbol list, we recommend that you read the latest official documentation mentioned above or visit this website. There are different standards and notations depending on the type of process you are describing. Such different standards of notation and symbolism of diagrams are:

- Class Diagrams
- Composite Structure Diagrams

- Package Diagrams
- Component Diagrams
- Deployment Diagrams
- Object Diagrams
- Profile Diagrams
- Use Case Diagrams
- Activity Diagrams
- State Machine Diagrams
- Sequence Diagrams
- Communication Diagrams
- Interaction Overview Diagrams
- Timing Diagrams

Note that the above list is not exhaustive. Some companies also have their own recognised UML standards such as Cisco. We will not be covering every single one of the diagrams listed above, but only the most useful ones. There are two types of diagrams in UML.

Definition 4.1. Structural Diagrams

Structural diagrams focus on illustrating the organisation of a system by depicting its components. The goal is to represent elements that make up the system and the relationships between them. These include composite structure, deployment, package, profile, class, object and component diagrams.

Definition 4.2. Behavioural Diagrams

Behavioural diagrams focus on illustrating the dynamic aspects of a software system, showcasing how it behaves, responds to stimuli and undergoes state changes during runtime. These include activity, use case, state machine, sequence, communication, interaction overview, and timing diagrams.

4.2 Class Diagrams

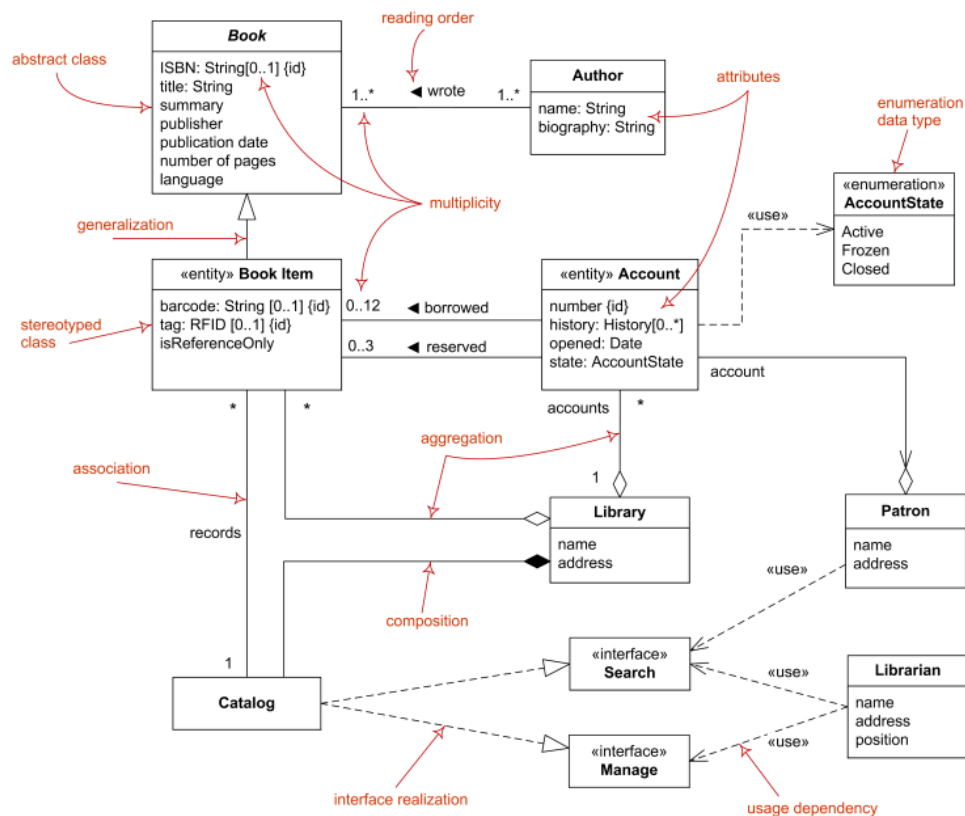


Figure 11: Class Diagrams Example

UML Class diagrams illustrate the structure of the system using the classes defined. The name of the class is written above, with its attributes below the name. Below attributes we find the methods. The return type is also state in both attributes and methods. Class visibility is shown using symbols + (public), - (private), (protected), ~ (package local). You can also describe the classes using their relationships, i.e. association, inheritance, realisation, dependency, aggregation and composition.

4.3 Composite Structure Diagrams

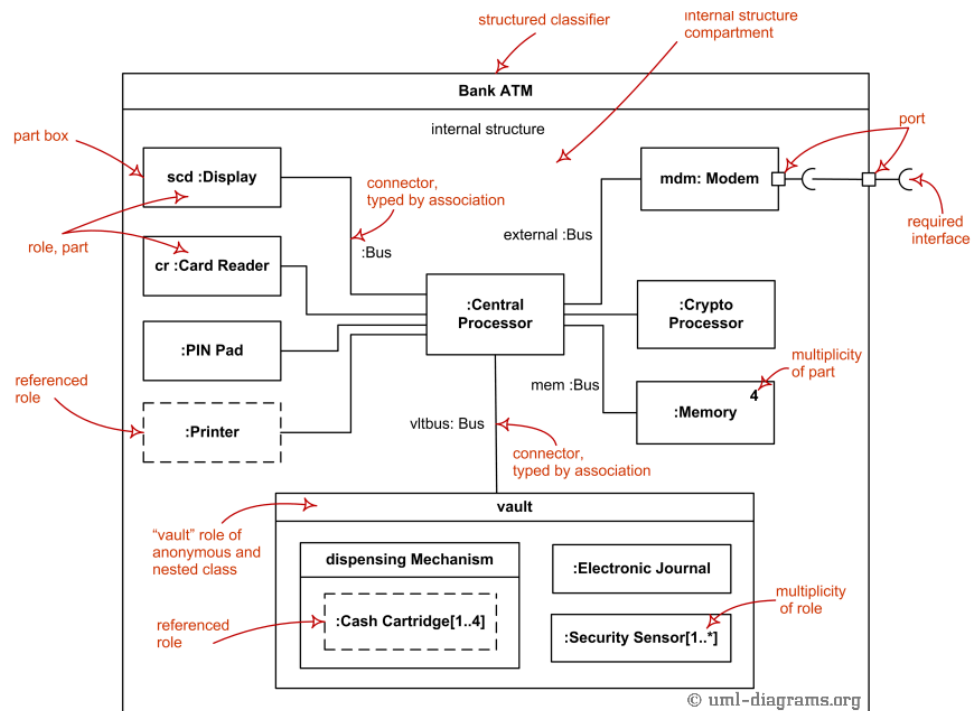


Figure 12: Composite Structure Diagrams Example

A composite structure diagram can be used to show the internal structure of a classifier, classifier interactions with environment through ports or a behaviour of collaboration. Similar to a class diagram, except it allows the user to fully explain the internal structure of multiple classes and show the interaction between them. In other words, composite structure diagrams show the internal parts of a class.

4.4 Package Diagrams

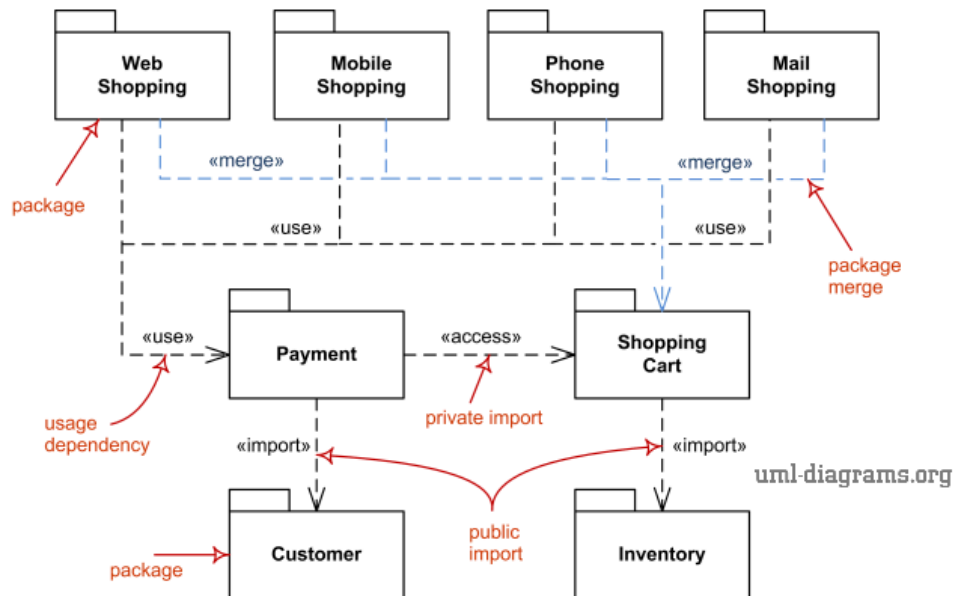


Figure 13: Package Diagrams Example

Package diagrams illustrate the location and organisation of model elements in a medium scale or large scale projects. It can display both the structure and the dependencies between subsystems or modules. Packages can be imported, merged, accessed or used. You can alternatively divide a group of packages to their own subsystem for better representation.

4.5 Component Diagrams

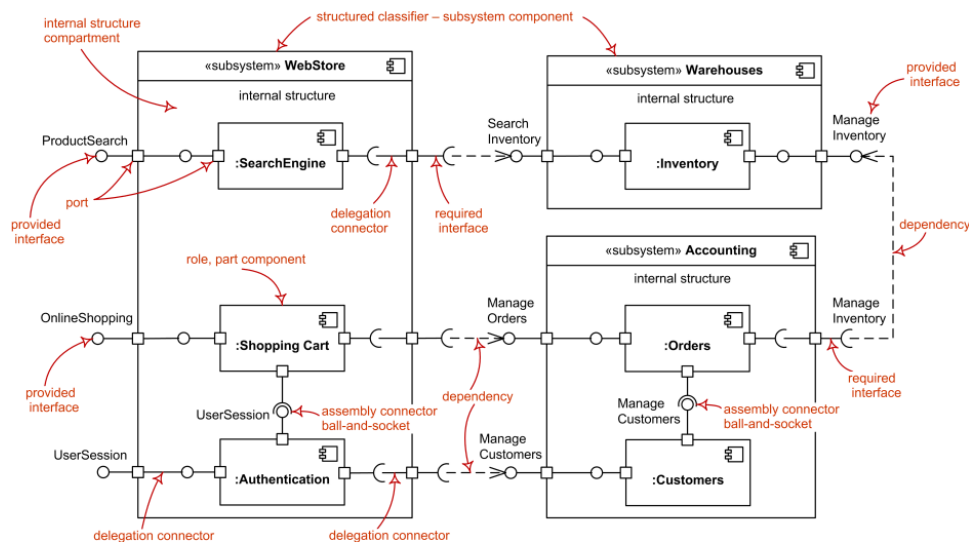


Figure 14: Component Diagrams Example

Component diagrams are used for modelling physical aspects of object oriented systems that are used for visualising, defining, and documenting component-based systems. They are similar to

class diagrams, with focus on system components that are used to model the static implementation view of a system. It is most typically used in component-based development. A component diagram would show how a web app's frontend component connects to a backend API component, which then connects to a database component, providing a high-level abstraction and general description of a system.

4.6 Deployment Diagrams

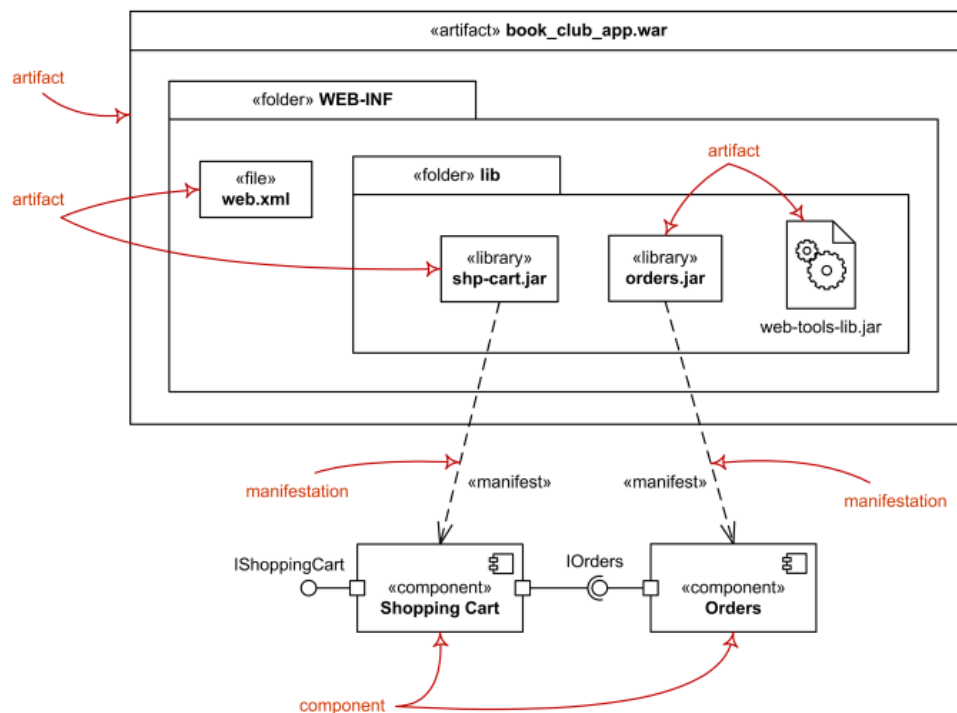


Figure 15: Deployment Diagrams Example

Deployment diagrams are used for modelling components and relationships between components and classifiers with deployment artifacts to deploy targets. It helps represent the static representation of a system deployment. Deployment diagrams model the elements of physical hardware or software and ways of communicating between them. It can be used to plan system architecture or documenting deployment of software components or nodes. It is a special kind of class diagram that focuses on system nodes.

4.7 Object Diagrams

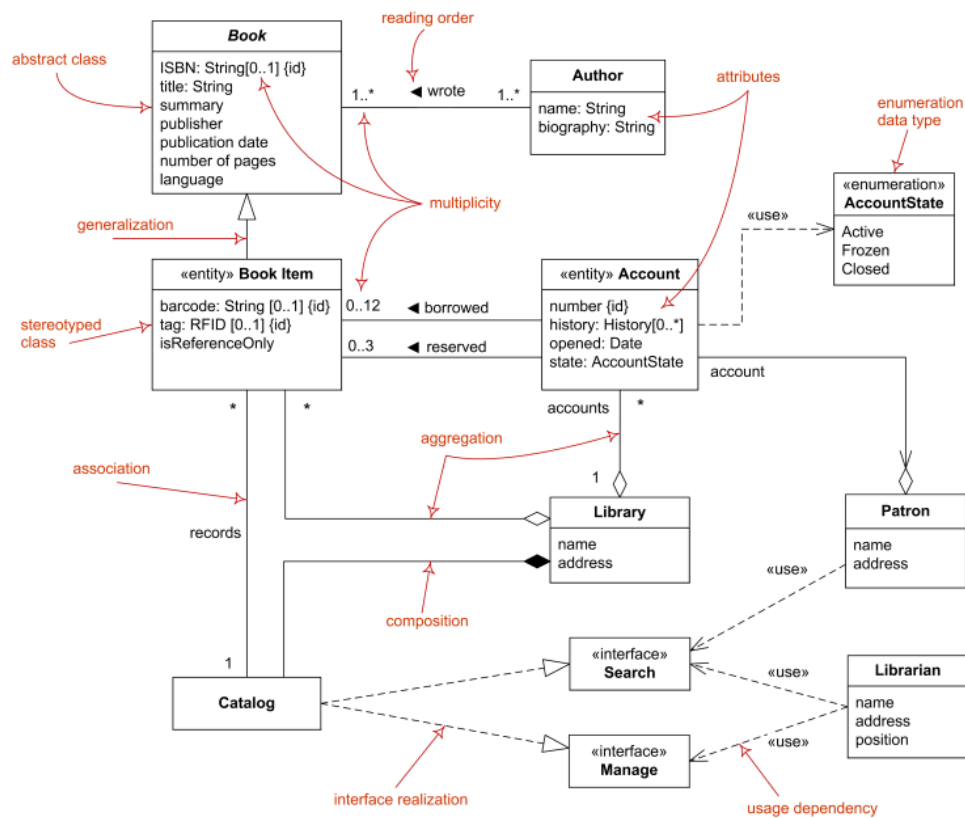


Figure 16: Class Diagrams Example (Object requires value substitution)

Object diagrams are similar to that of a class diagram. It captures a snapshot of the detailed state of the system objects and their relationship at a specific point in time. It is primarily used to explain complicated cases, or to test the robustness of the existing class diagram by substituting a scenario.

4.8 Profile Diagrams

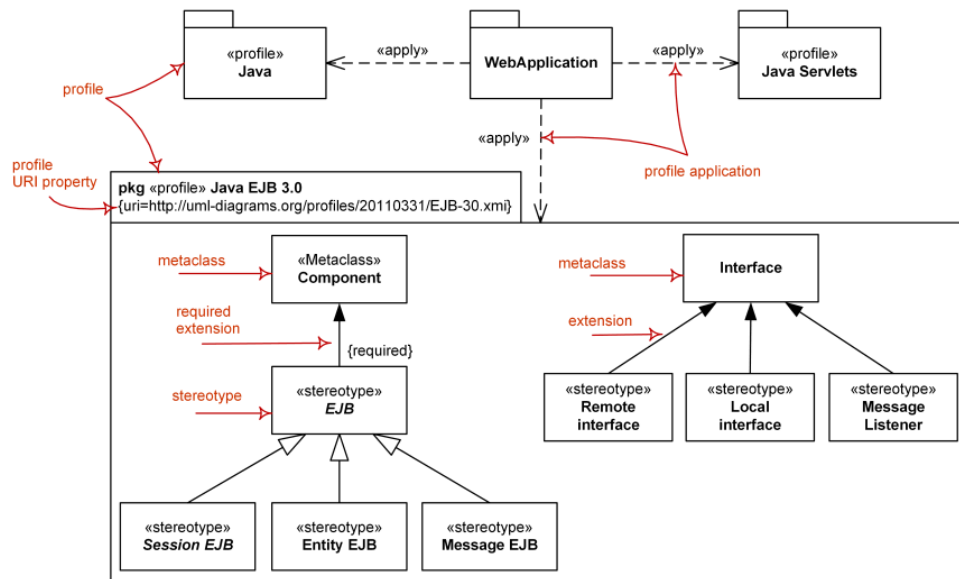


Figure 17: Profile Diagrams Example

Profile diagrams provide a general extension mechanism for customising UML models for specific domains and platforms. A profile is a set of extensions that jointly configure UML for a specific domain. An example of a specific platform is .NET. There are three major types of extensions, stereotypes, tags and constraints. Stereotypes let you create the vocabulary of UML. New model elements can be created from existing ones with more specific properties. E.g. **«button»**. Tags are used to create extra information under the names e.g. author or version.

4.9 Use Case Diagrams

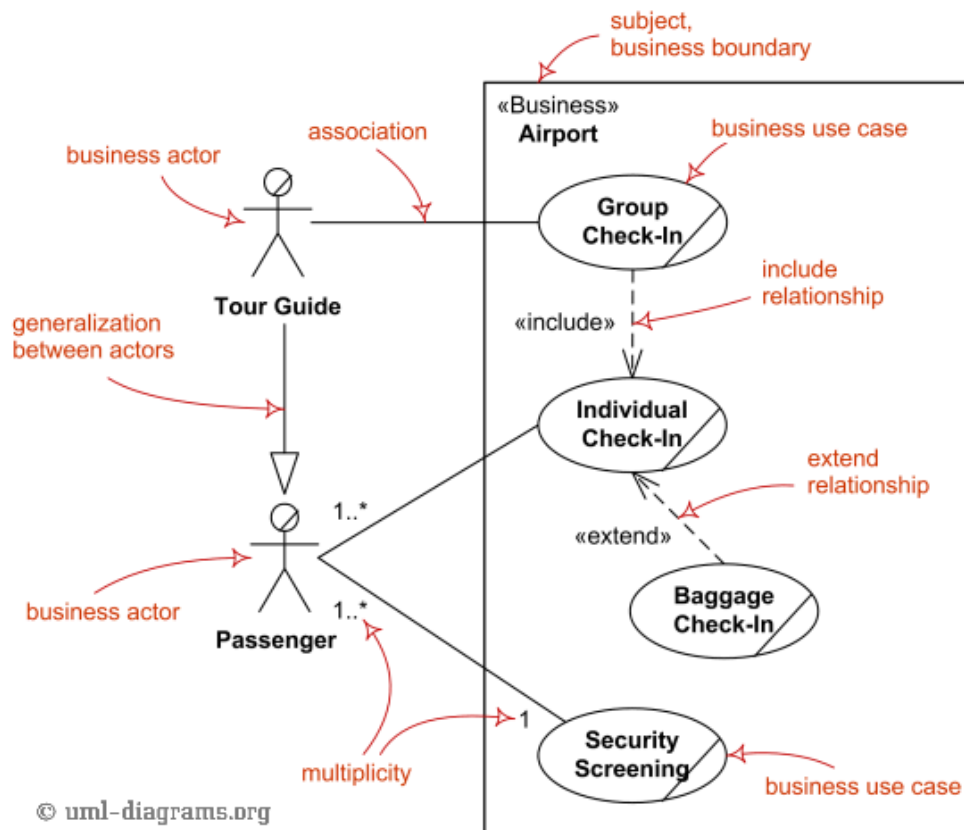


Figure 18: Ucase Case Diagrams Example

Use case diagrams summarise the relationships between uses cases, actors and systems. Use case diagrams do not the order in which steps to achieve the goals of each use case are taken. Uses cases only show functional requirements of a system. It is usually constructed as early stage of a development, to specify the context of a system, capture requirements, validate system architecture and drive implementation and generate test cases.

4.10 Activity Diagrams

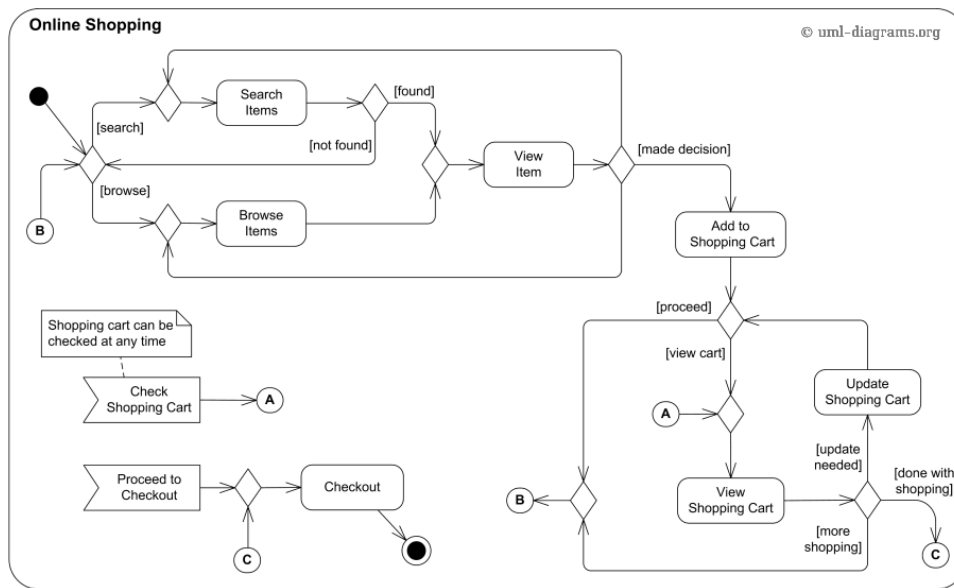


Figure 19: Activity Diagrams Example

Activity diagrams demonstrate the object flow with emphasis on the sequence and conditions of the flow. It is an extended version of flowchart that models the transition from one activity to another, and shows how system activities are coordinated to provide a service that can be at a different level of abstraction.

4.11 State Machine Diagrams

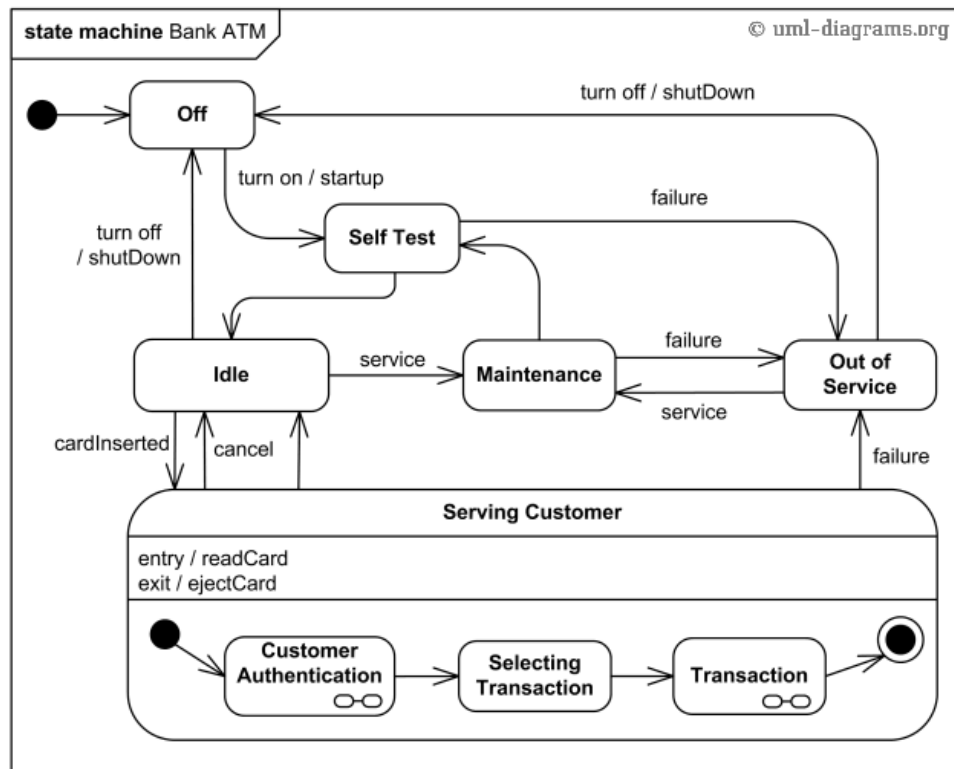


Figure 20: State Machine Diagrams Example

State machine diagrams show discrete behaviour of a part of design systems through finite state transitions. It can also display how state of objects can be changed depending on the behaviour. It can be applied to objects or anything that can change state e.g. actor.

4.12 Sequence Diagrams

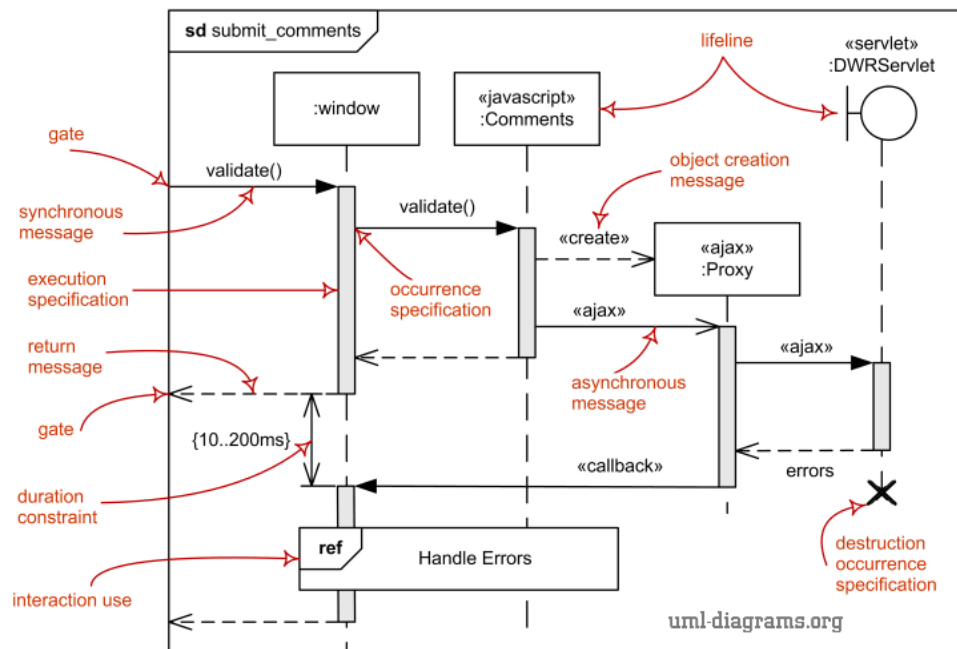


Figure 21: Sequence Diagrams Example

The most common type of interaction diagram, it focuses on the message interchange between a number of lifelines. It details how operations are performed. The horizontal axis displays elements, whereas the vertical axis displays time. Note that the time axis displays the sequence of order, and not the duration it takes for actions.

4.13 Communication Diagrams

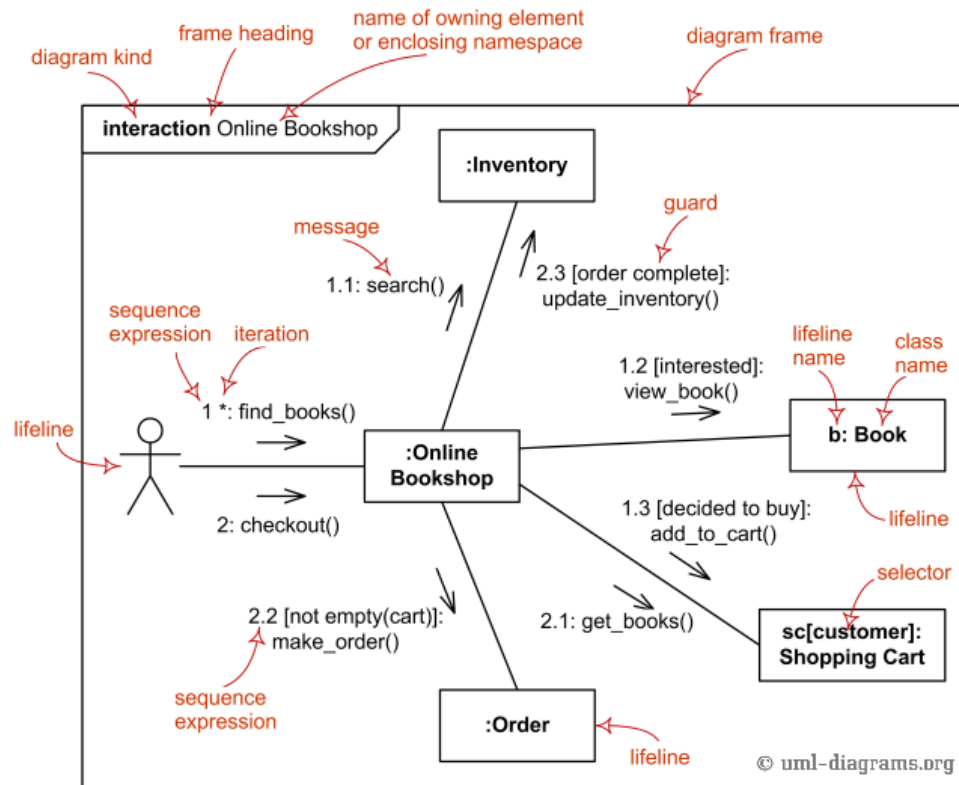


Figure 22: Communication Diagrams Example

Also called the collaboration diagram, illustrates interactions between objects and/or parts using sequences messages in a free-form arrangement. The purpose of this diagram is to illustrate the messages or roles that each object has in the system. It captures how systems work with each other using systems without specifying time or specific cases.

4.14 Interaction Overview Diagrams

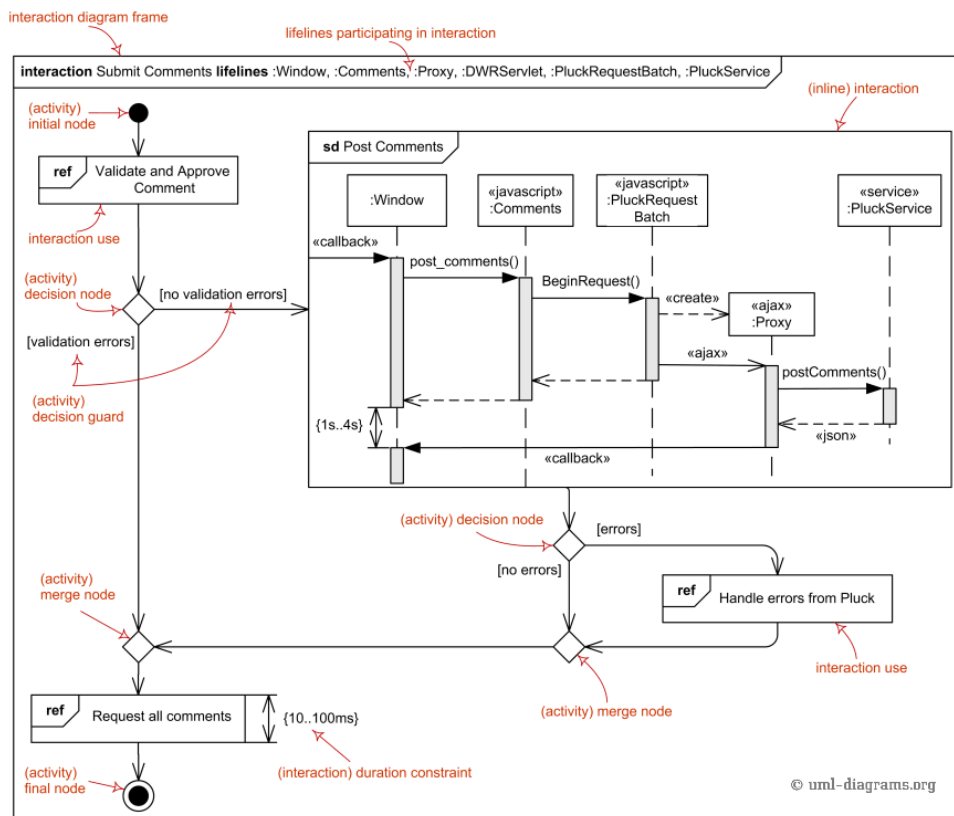


Figure 23: Interaction Overview Diagrams

These diagrams are used to provide overview of the flow of control where nodes of the flow are interactions or interaction uses. It is a high-level abstraction that illustrates the flow of activity between diagrams. It resembles an activity diagram which the elements were replaced by small sequence diagrams.

4.15 Timing Diagrams

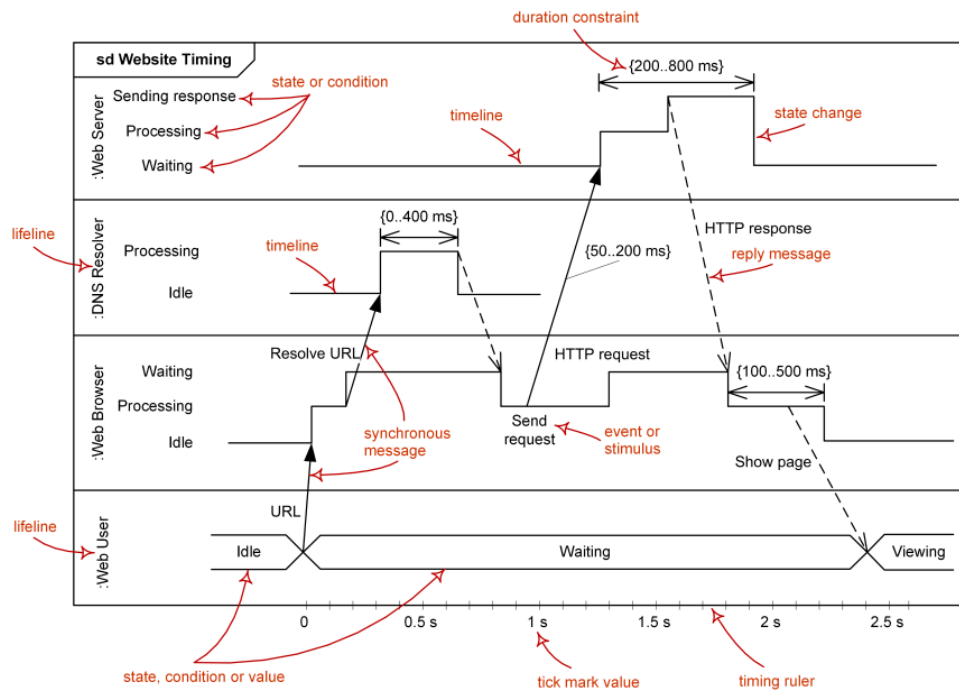


Figure 24: Timing Diagrams Example

Timing diagrams are used to show interactions when a primary purpose of the diagram is to reason about time. Timing diagrams focus on conditions changing within and among lifelines along a linear time axis. It indicates the intervals between state changes of various objects.

5 Communication & Software Presentation

5.1 What is Communication

Definition 5.1. Communication

Communication is the act of transmitting or receiving information.

Communication is an essential skill to ensure that the information received and transmitted has been provided with clarity, understanding and unambiguity. To enhance our understanding of communication, several scientific models were created. One of such models is the Shannon and Weaver transmission model developed in 1949, illustrated below.

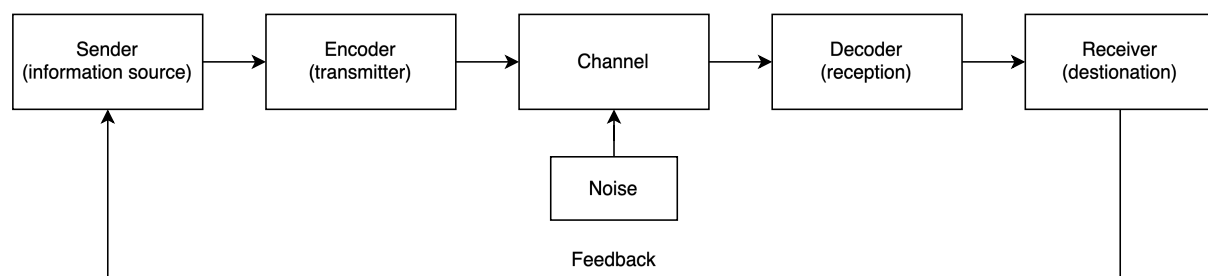


Figure 25: Shannon and Weaver Transmission Model

The Shannon-Weaver model explicitly states that there is a source, a transmitter, a channel, a receiver and a destination. In the channel, there may be a level of noise and interference added.

Another proposed model is Wiio's Law of (mis-)communication. Developed by Osmo Antero Wiio, he is best known for his humorous quote "*Communication usually fails, except when it is a mistake*".

- Communication usually fails, except when it is a mistake.
- If communication can fail, it will fail.
- If communication cannot fail, it still usually fails.
- If communication seems to succeed in the way you intended, someone is misunderstood.
- If a message can be interpreted in several ways, it will be interpreted in a way that maximises damage.
- There is always someone who knows better than you what your message means.
- The more we communicate, the more communication fails.

In other words, what you communicate and have in your mind will never have the same message the moment you decide to share it. As such, there is a requirement for more complicated models, utilising the idea of pattern matching. This module as such explores the idea of a bottom-up processing and top-down processing. The brain does not recognise objects instantly. Instead, it looks for basic features like shape and colour. Different brain networks detect these features independently and in parallel. Bottom-up processing, for example, recognises the shape of letters. Top-down processing combines those shapes into words you understand. Neocortex is responsible for top-down processing, whereas bottom-up uses more primitive parts such as brainstem and cerebellum.

5.2 Three Principles of Communication

To understand "what effect am I having?" when we relay information to other people, we need to delve deeper into the pattern-matching model of communication. The pattern-matching model of communication suggests three important principles.

1. Communication is continuous. If we are always updating our understanding, then communication must be continuous to be effective. It cannot be a one-time event, like a radio broadcast, but a process.
2. Communication is complex. Whatever we understand is communicated, including everything we observe and not just listening to the words that someone says. Body messages, signals, including the "music" of their voice and the "dance" of their body.
3. Communication is contextual. That is, communication never occurs in isolation. The meaning of communication can be influenced by at least five different contexts such as psychological, relational, situational, ecological, cultural, etc.

And as such, we propose a new definition of communication.

Definition 5.2. Communication

Communication is the processing of creating common understandings.

5.3 Three Levels of Understanding

We propose three levels of measuring and understanding understanding.

- **Relationship** - We create an understanding through verbal, vocal and physical behaviours. It is important to understand the intention of the communication and the reason. Other people should not feel excluded or interrogated, so avoid talking too much about yourself and asking other people direct questions about themselves. Relationships ensure that when communicating, individuals are in the same rhythm.
- **Information** - Once a relaxed connection is established, we are ready to share information. For example, expressing "I see" or "ah" exclaims information that we gain an understanding of a topic. However, we often misunderstand each other due to approaching a problem from different angles. Information is always "out there", always in our minds. It is dynamic.
- **Action** - we communicate to encourage an action. The key to effective action is not accurate information, but compelling ideas. Ideas give meaning to information.

5.4 Data and Information

It is important to note the difference between data and information. A research begins with the process of collecting data. In general, data refers to facts or statistics collected by a researcher for analysis in its original form. When data is processed, it is transformed in such a way that it becomes useful to the users, known as "information. Data is unsystematic fact or detail about something, whereas information is systematic and filtered form of data that is useful. A data can either be qualitative (non-numerical) or quantitative (numerical), that can be classified to primary(in-person) or secondary (learned from someone else). Or even internal compared to external data.

Definition 5.3. Data

Data is a collection of facts and details such as text, figures, observations, symbols, numbers, or even descriptions of a thing, event or object, collected with a view to drawing conclusions.

Definition 5.4. Information

Information is form of data that is processed, organised, specific and structured form of data presented in a given setting. It assigns meaning and improves the reliability of data, thereby ensuring understandability and reducing uncertainty.

The key differences are:

- Data is simple text and numbers while information is processed and interpreted data.
- Data is unorganised form i.e. they are randomly collected facts and figures that are processed. Information is organised which presents data better and adds meaning.
- Data is based on observations and records stored in computers or remembered by a person. Information is concluded through analysis, not given as an immediate output.

- Data is bare and is raw facts. Information refers to facts about a particular event or subject that is refined through processing.
- Data is not always relevant, whereas information is always relevant.
- Data does not depend on information. Information depends on data.
- Data is not always useful, whereas information is after the process is.

5.5 Three Ingredients of a Presentation

There are three main ingredients that must be present for a presentation. Without any of these ingredients, the presentation will fail. These are:

- The presentation itself
- The audience
- The presenter

Unfortunately, presentations are also an extremely expensive way to get your message across. They take up extra time compared to other methods, which is money. Therefore, one must be sure that the presentation is worth the effort. Good reasons include:

- Communicating time-critical information to a large group of people
- Persuading audience to make a choice, change their decision, take a course of action or convey information to others in person
- Audience is interested, concerned or needs to hear what you have to say
- Teaching a skill or information to more than three people at a time
- Have a clear set up of objectives
- Acts as a bonding exercise for the people participating
- Have the time, energy and commitment to make the presentation

And before preparing a presentation, it is important to ask yourself:

- Is there a better way to convey this information?
- Do I have all the information I need?
- How much will this presentation cost and is it cost-effective?
- What is this presentation about? What will the audience do, think and feel after hearing the presentation?

There are multitude of benefits to information, such as:

- Save time and/or effort
- Make people feel comfortable

- Improve their health
- Make them unique
- Help them gain control
- Win a reward
- Attract the opposite sex
- Help protect their reputation

5.6 How To Start a Presentation Effectively

There are different methods of starting a presentation, but only few of them are truly effective. The start of a presentation is critical: it creates the first impression and attracts the reader's attention for the rest of the presentation. The methods of effectively starting a presentation are:

1. Starting with introduction - you can use a memory re-wiring formula such as a good story, who you are, why the things you say matter, to create an emotional connection. You can also do a stereotypical introduction from a basic template, including a stereotypical joke about your position.
2. Humour - surprise the audience right away with something they wouldn't expect to hear. Start the presentation with a joke, everyone loves an unexpected twist like a door bell ring. You await the result. It can be a provocative statement, or a rhetorical thought-provoking question. It can be a bold number, factor, condition.
3. Captivating visualisation - you can use visualisations to make a bold statement at the beginning. Visuals are powerful, our brains only need 13 milliseconds to process what our eyes see. Visuals with statistics are also effective.
4. What if...? - give the audience a sense of what will happen if they decide to listen to you and follow your advice. Alternatively, you can use "imagine", "visualise" or "think about".
5. Curiosity - use instant attention click-bait like title grabbers. Explicitly state that you will say something you don't know, and that you're eager to change that. You can tell a story, pause in the middle, and delay the conclusion. Withhold key information, but not for too long.

5.7 Elements of an Effective Presentation

Your job as a presenter is to turn complex technical information into a cohesive and a compelling story, and then present that story in an engaging way. To accomplish this task, you need to consider what you are saying and how you are saying it. As such, you require a clear strategic plan to ensure the effectiveness using the key elements.

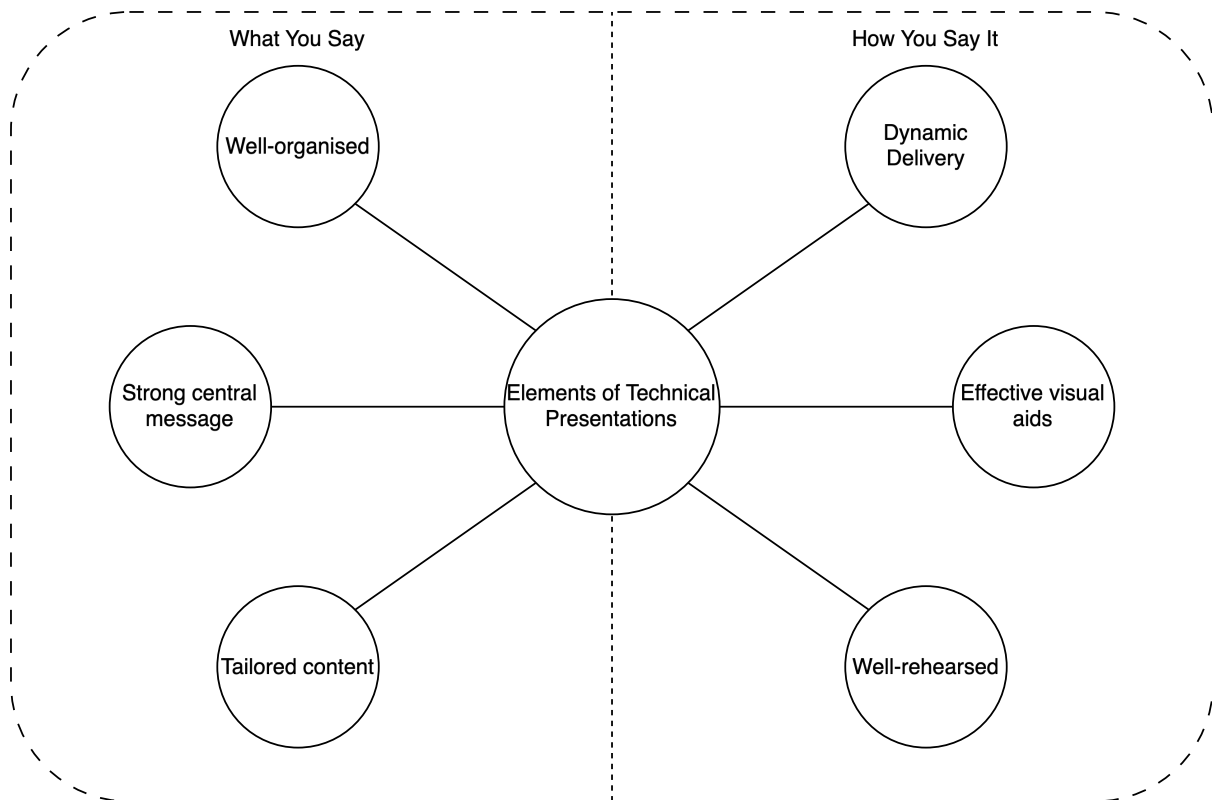


Figure 26: Elements of an Effective Presentation

And these key elements are:

- **Well-organised** - presentation must follow a logical way. Start broad and narrow to key technical details, ending by reconnecting those details to a bigger picture. You can think of your presentation as a story. Creating and resolving tension through questions or problems helps the audience care and remember your message.
- **Strong central message** - the message should be driven by your strategic goal and strongly supported by details in your presentation. Any details that do not relate to the central message can be cut or moved to "additional slides" to keep it on topic.
- **Tailored content** - make sure the content is relevant and tailored to answering your central message. You should adjust the overall level of detail, technical sophistication such as technical jargon and how the topic is framed. Make it relevant and digestible.
- **Dynamic delivery** - make your information dynamic and engaging, both through verbal and non-verbal communication. Make eye contact and scan the room for inclusion of audience. Be mindful of your pacing.
- **Effective visual aids** - simple, clean visuals can help you get your point across faster. Visuals should add value and directly support the point that you are trying to make on a particular slide.
- **Well-rehearsed** - rehearse your presentation by studying the timing. There will be a time limit. Ensure you cover the right amount of content. Practice it out loud, do not rush. Refine your delivery, make the audience feel comfortable.

Additionally, you can use the TAG method, which is the type, audience, goal method. In other words, you answer:

- What type of presentation is it?
- Who is the audience?
- What is your goal?

5.8 10 Tips for a Successful Presentation

1. **Prepare Well** - This may seem obvious, but thorough preparation is crucial for your presentation. Above all, practise the sequence so you can approach the actual presentation with confidence.
2. **Don't Memorise the Entire Presentation** - While you should be well-prepared, memorising everything word-for-word can be a disadvantage because if you forget a word, you might panic. The important thing is to know what you want to say for each slide or section.
3. **Work with a Clear Structure** - Ensure your presentation is clearly and cohesively structured. This helps you avoid confused faces in the audience and makes you feel more confident because you can easily navigate through your presentation's clear structure.
4. **Don't Write Everything You Want to Say** - Avoid writing everything you want to say on the slide. Stick to manageable bullet-point content and expand on it during your presentation, which makes it easier to follow.
5. **Think About Your Body Language** - If you fidget constantly, you'll automatically feel more nervous. Try to remain calm and adopt a confident, open posture with your hands out of your pockets.
6. **Involve the Audience** - Depending on the topic and situation, it can be a good idea to engage the audience in your presentation. Framing content as questions not only makes your presentation more relaxed but can also reduce your stress and generate more interest.
7. **Give Yourself Time** - Try not to put yourself under time pressure. You should know approximately how long you need, but rushing through so that no one understands anything doesn't help. Speak calmly and clearly—this is more comfortable for both you and your audience.
8. **Checking Something Isn't the End of the World** - No one expects absolute perfection from you. The occasional glance at the slide or your notes is completely normal and acceptable. However, it's important not to lose contact with your audience or give the impression you're reading from notes.
9. **Don't Stand Fixed in One Place the Whole Time** - If you're nervous, it can help to move around a bit during your presentation. When moving to the next slide, for example, you can move to the other side of the screen or take a quick sip of water. It's important to strike a balance—some movement is good, but avoid constantly pacing back and forth.
10. **Be Open and Honest** - Of course, even with the best preparation, you can stumble. If you lose your thread, it's advisable to be honest. The audience will usually understand if you pause briefly to gather your thoughts, and if you speak openly about problems during your presentation, someone can often help you.

5.9 8 Tips for Reducing Anxiety

1. **Organise** - A lack of organisation is a primary cause of anxiety, so ensuring your thoughts are well-structured is essential. Knowing you are prepared gives you the confidence to focus your energy entirely on the delivery.
2. **Visualise** - Imagine entering the room and delivering your presentation with enthusiasm and confidence. Mentally rehearsing every detail of the situation helps you focus on the specific actions required for success.
3. **Practise** - Stand up and use your visual aids as if an audience were present rather than just rehearsing in your head. It is recommended to record yourself or seek feedback to make necessary changes before your final rehearsal session.
4. **Breathe** - When your muscles tighten and you feel nervous, you may not be breathing deeply enough. The first thing you should do is sit upright yet relaxed and take several deep breaths.
5. **Focus on relaxing** - Instead of thinking about the pressure, concentrate on the act of relaxation. Repeat the phrase "I am relaxed" in time with your breathing for several minutes to clear your mind.
6. **Release tension** - Nervous energy can become trapped in your limbs, causing your hands or legs to shake. Perform simple isometric exercises by tensing your muscles from your toes to your fists and then suddenly releasing them to discharge this energy.
7. **Move** - Speakers who stand perfectly still often experience more tension, so allow your muscles to flex and use natural gestures. Moving towards the audience or shifting your position helps release physical stress and keeps the listeners engaged.
8. **Make eye contact with the audience** - Deliver your presentation to one person at a time, connecting with members of the audience as individuals to make the experience more personal. This contact helps you relax by reducing feelings of isolation and allowing you to respond naturally to their interest.

5.10 Personal Appearance

For women:

1. **Clothing** - Attire should fit well without being too tight, with hem lengths chosen to ensure a professional appearance when seated on stage. Longer sleeves are generally recommended to maintain a formal, businesslike look during your presentation.
2. **Colours and fabrics** - Select two or three colours that complement your skin tone and hair, pairing them with complementary accessories for variety. Opt for high-quality fabrics that do not make noise when you move, and avoid stark colours like bright orange or white that might distract from your face.
3. **Jewellery** - Avoid items that sparkle, dangle, or make noise, as subtle accessories are more appropriate when you are the presenter. Distracting earrings or bracelets can annoy the audience and take their focus away from the content of your speech.

4. Makeup - Application should be simple and complementary, as excessive makeup can draw negative or unwanted attention. When done well, it controls oily areas that reflect light and helps you appear more composed even in difficult presentation situations.
5. Hair - Your hairstyle should contribute to a positive overall impression without becoming the most dominant feature of your face. While styles are highly individual, they should remain understated to ensure the focus remains on your message.

For men:

1. Suits - Suits should be well-tailored and are best chosen in classic colours like navy, grey, or black. Avoid patterns, bright colours, or baggy styles to ensure you project a professional and reliable image.
2. Jackets - Unlike many styles for women, men's suit jackets are specifically designed to be buttoned. Depending on the formality of your presentation, you may choose to button your jacket, leave it unfastened, or remove the coat entirely.
3. Shirts - Select well-fitting shirts in subtle colours that are not overly bright. If you are prone to sweating, wearing a cotton T-shirt under a white shirt can help, though light grey is preferred over white for television appearances.
4. Ties - Use ties to complement your eye colour and complexion rather than opting for a distracting traditional red. Subtler tones are often more effective as they allow the audience to focus on you rather than your accessories.
5. Footwear - Your shoes should be appropriate for the occasion, comfortable, and well-polished. Always ensure your socks match your attire and are long enough to cover your skin when you are seated.
6. Grooming - Hair and facial hair should be well-groomed to frame the face neatly, regardless of the specific style. Moustaches should be carefully trimmed above the lip line to maintain a tidy and professional look.

5.11 10 Tips for Planning Successful Slides and Visualisation

1. Use slides sparingly - Overusing slides is a primary pitfall in technical presentations. A helpful rule of thumb is to aim for one slide for every two minutes of presentation time to maintain audience engagement.
2. Make slides pictorial - Use graphs, equipment photographs, and diagrams to give the viewer a visual insight that would otherwise require many words or columns of figures. These elements help the audience grasp complex information much faster than text alone.
3. Present one key point per slide - Keep the focus of each slide simple and clear to ensure maximum impact. Presenting more than one main idea at a time can seriously diminish the effectiveness of your message and confuse the audience.
4. Make text and numbers legible - Ensure your font size is at least 20pt so that everyone in the room can read the content easily. If the data is too dense, be prepared to provide extra explanations in a handout or highlight specific areas of a diagram.

5. Use colours carefully - Limit yourself to no more than three or four colours per slide to avoid a cluttered and unprofessional appearance. Choose high-contrast combinations, such as light yellow text on a navy blue background, to ensure everything is easy to see.
6. Ensure visuals are large enough - Always check your presentation from the very back row of the room where the audience will be sitting. This ensures that every element on your slide is clearly visible to those furthest from the screen.
7. Use graphic data - Avoid using tables whenever possible and opt for graphs instead. Graphs allow the viewer to visualise data and trends in a way that lists of numbers simply cannot achieve.
8. Make images and diagrams easy to view - Ensure that all photographs and diagrams are clear and that any labels are legible even from the back of the room. Visuals that are difficult to see from a distance will only distract and frustrate your listeners.
9. Avoid unnecessary slides - If a point can be made simply and verbally, such as a basic title or a brief statement, there is no need for a dedicated slide. Minimising clutter helps keep the audience's attention on you as the speaker.
10. Use animations sparingly - Complex builds and animations should be used very selectively as they can often interfere with the substance of your message. If used excessively, they become a distraction rather than a helpful tool.

5.12 15 Tips for Creating Better Slides

1. Allow two minutes per slide - Use this rule of thumb to estimate the total number of slides needed for your presentation. If your slides include complex diagrams or detailed explanations, you should allocate even more time to each one.
2. Use functional headings - Whenever possible, your slide titles should contain the conclusion you want the audience to reach or the specific action you want them to take. This ensures the main message is immediately clear to every viewer.
3. Follow the 5x5 rule - Limit the content of each slide to a maximum of five bullet points and no more than five words per point. You should also maintain a consistent grammatical structure and style for every item on the slide.
4. Use phrases and keywords - Select your words carefully to communicate the essence of each point quickly without using full sentences. If long descriptions or detailed explanations are necessary, save that information for your printed handouts.
5. Use sentence case - Capitalise only the first letter of the first word and any proper nouns within your bullet points. Using capital letters for every word can make a slide look cluttered and much harder for the audience to read.
6. Avoid building every slide - Revealing bullet points one at a time can be effective, but the audience will grow tired of the repetition if you do it on every slide. Stick to a single type of transition effect throughout your presentation to maintain professionalism.

7. Number every slide - Adding slide numbers helps viewers stay on track if they lose the thread of the discussion or join the session late. It also makes it easier for the audience to refer back to specific slides during a question and answer session.
8. Stick to a style guide - Follow your company's colour scheme and use dark backgrounds with light text for maximum legibility. For large audiences, white or yellow text on a navy blue background is often the easiest combination to read.
9. Consider sans serif fonts - Use fonts such as Arial or Helvetica, as many people find them easier and faster to read when projected. These styles significantly improve legibility, particularly for those sitting at the back of a large room.
10. Use a large font size - Aim for a minimum font size of 24 points unless you are presenting to a very small group. You should never expect an audience to be able to read a font that is smaller than 20 points.
11. Use images and videos - Incorporating photographs and videos can help break the monotony of text-based slides. However, ensure that any drawings or diagrams you include remain simple, accurate, and relevant to your message.
12. Limit the use of clip art - Use clip art very sparingly or, preferably, avoid using it altogether. Many modern professional environments discourage its use as it can detract from the credibility of your presentation.
13. Animate drawings selectively - Building a complex drawing or process in stages is a great way to maintain audience attention and explain flow. Be careful not to overdo it, as excessive animation can make the audience focus on the visual tricks rather than your speech.
14. Have a backup plan - Always carry a backup version of your presentation in case you encounter technical difficulties with your laptop or the projector. Being prepared for equipment failure ensures you can stay calm and continue your delivery.
15. Remember you are the messenger - Your slides are merely a communication tool and should not be the primary focus of the presentation. You are the one carrying the message, and the visuals are only there to support what you are saying.

5.13 Layout

There are general layout rules for a presentation that you must follow during speaking.

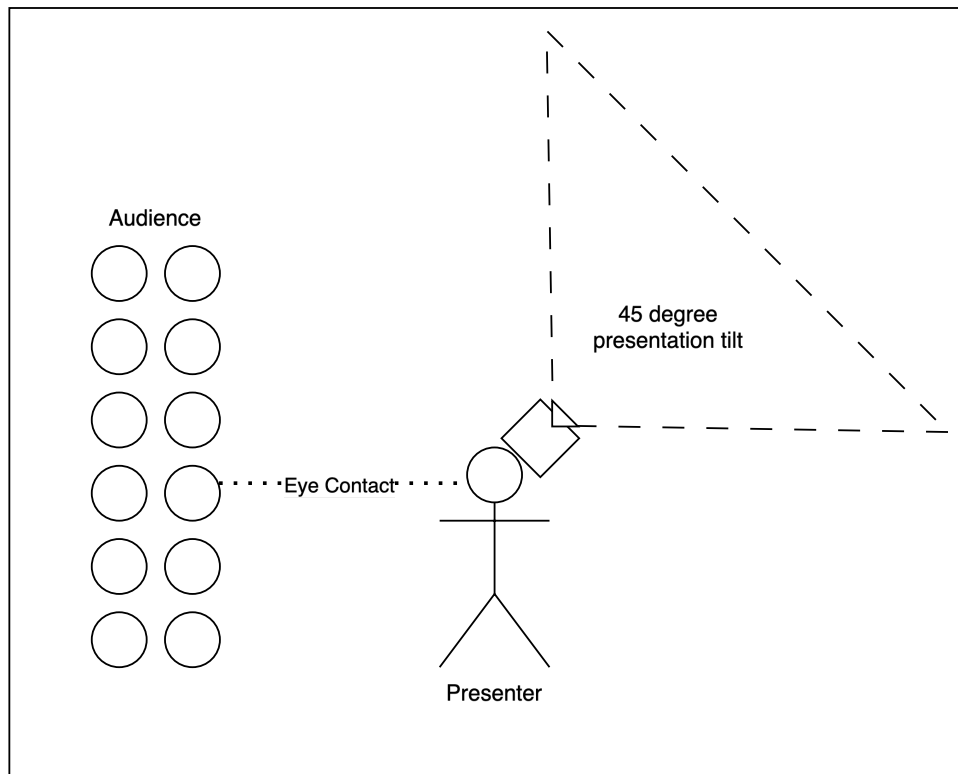


Figure 27: Layout of a professional presentation

- Create eye contact with the audience, not the presentation itself.
- The presenter should be in a central position, as such, tilting the presentation is the best practice.
- Do not put your arms in a fig leaf position, wring your hands nervously, cross your arms, keep your hands behind you like handcuff, or keep your hands in your pocket.

5.14 Teleconferencing and Videoconferencing

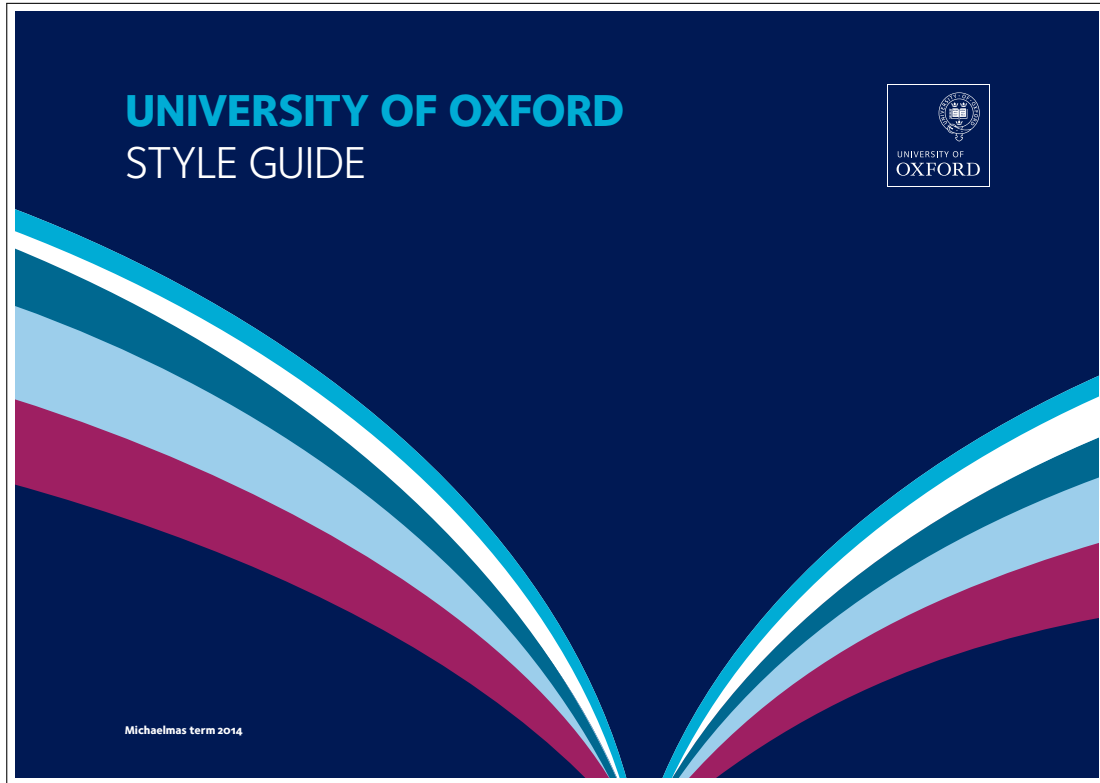
Virtual meetings have become a staple of modern business, offering a flexible alternative to traditional face-to-face interactions. While these technologies provide significant logistical advantages, they also introduce unique challenges regarding participant engagement and communication clarity.

Table 8: Advantages and Disadvantages of Virtual Conferencing

Aspect	Description
Distraction (Conference)	Participants often become distracted because there is no visual element to hold their attention. Without a screen to focus on, it is much easier for the mind to wander during the discussion.
Communication (Conference)	Voice is the only method for engaging participants, meaning there is no opportunity for non-verbal communication. This makes it harder to convey tone or read the room effectively.
Loudspeaker Syndrome	Using a loudspeaker creates a situation where you can never be certain who is actually listening or paying attention. This lack of accountability can undermine the effectiveness of the meeting.
Phone Etiquette	Poor etiquette is common as people may act rudely when they feel invisible to the rest of the group. The perceived anonymity of a phone call often leads to less professional behaviour.
Equipment Misuse (Video)	Technical benefits are often lost when equipment, such as wide-angle cameras, is not configured correctly. This makes it difficult for participants to see exactly who is speaking at any given time.
Communication Friction (Video)	Unrealistic expectations for clear communication can lead to frustration when voice-activated microphones make dialogue feel awkward. These technical limitations can hinder the natural flow of a two-way conversation.
Cost Savings	Both methods offer significant financial benefits by eliminating the need for expensive airfare, hotels, and meals. These savings can amount to thousands of pounds over the course of a business year.
Time Efficiency	Meetings can be held directly from your workplace, minimising the time spent away from your primary duties. Strict time constraints on equipment also ensure that sessions start and end on schedule.
Enhanced Teamwork	Virtual meetings allow for greater participation as more staff members can join without the limitations of travel budgets. This allows information to be shared and refined by the whole team in real time.

Appendices

A Style Guide Example



UNIVERSITY OF OXFORD

STYLE GUIDE



Contents

1 Introduction	Objectives of the style guide 1 How the guide is arranged 1 How to use the guide 1 What is/is not included in the style guide 1 Quick reference guide 1	17 Names and titles	General titles 17 Oxford-specific titles 17 Other titles 18 Combining titles 19 Postnominals 20
2 Abbreviations, contractions and acronyms	Abbreviations 2 Contractions 2 Acronyms 2 Specific abbreviations 3	21 Highlighting/emphasising text	Bold 21 Italic 21 Underlining 21
4 Capitalisation		22 Word usage and spelling	Common confusions in word usage 22 Spelling 23
7 Numbers	How to write numbers 7 Times 7 Dates 8 Spans of numbers and years 8	25 Miscellaneous	Personal pronouns 25 Plural or singular? 25 Addresses, phone numbers, websites etc 25
9 Punctuation	Apostrophe 9 Brackets 10 Bullet points 11 Colon and semicolon 11 Comma 12 Dashes and hyphens 13 Ellipsis 15 Full stop, exclamation mark and question mark 15 Quotation marks 16		

Introduction

The Oxford University Style Guide aims to provide a guide to writing and formatting documents written by staff on behalf of the University (or one of its constituent departments etc). It is part of the University's branding toolkit (www.ox.ac.uk/branding_toolkit) which enables the University's formal documentation to be presented consistently across all communications.

The style guide is not intended for public or external use, and does not purport to compete with OUP's professional writing guides and dictionaries.

Objectives of the style guide

We have three main objectives in writing this style guide:

- to provide an all-purpose guide to consistent presentation for University staff in written communications
- to review the guide at least once a year, ensuring that it properly reflects modern usage and is fit for purpose, and to update it as required
- as part of the review process, to invite proposals from members of the University who disagree with any existing guidance, and to act as an arbiter on those cases.

How the guide is arranged

The style guide is intended to be read as an interactive PDF, where it can be cross-referenced. However, the PDF can be printed if preferred for ease of reference.

If we update the style guide we will highlight on the main webpage (www.ox.ac.uk/styleguide) whether anything has changed as well as changing the term listed on the front cover.

How to use the guide

- search for a specific term (such as *semicolon*)
- browse through a section (such as **Punctuation**)

What is/is not included in the style guide

The guide does not tell you how to write. We aim to help you write correctly, and to encourage consistency across the University's written communications.

Quick reference guide

The general rule

If there are multiple (correct) ways of doing something, choose the one which uses the least space and the least ink. For instance:

- close up spaces and don't use full stops in abbreviations (eg 6pm)
- use lower case wherever possible
- only write out numbers up to ten and use figures for 11 onwards.

University of Oxford or Oxford University?

These terms are interchangeable and can either be alternated for variety or kept the same for consistency.

University branding information

Other information on University branding, including the use of the logo, can be found online at www.ox.ac.uk/branding_toolkit.

Queries

If you have any queries about using this guide, please contact:

Public Affairs Directorate
University of Oxford
Wellington Square
Oxford OX1 2JD

gazette@admin.ox.ac.uk

Abbreviations, contractions and acronyms

General rule

Don't use full stops after any abbreviations, contractions or acronyms and close up space between letters.

Abbreviations

These are formed by omitting letters from the end of a word.

- ✓ Medical Sciences → Med Sci
- ✓ Doctorate of Philosophy → DPhil
- ✓ ante meridiem → am
- ✓ post meridiem → pm

Contractions

These are formed by omitting letters from the middle of a word.

- ✓ Mister → Mr
- ✓ Doctor → Dr
- ✓ The Reverend → The Revd
- ✓ Saint → St
- ✓ Street → St

Acronyms

These are formed from the initial letters of words (whether the result is pronounceable as a word or as a series of letters) and should be written as a single string of upper-case letters.

- ✓ British Broadcasting Corporation → BBC
- ✓ Master of Arts → MA
- ✓ Acquired Immune Deficiency Syndrome → AIDS
- ✓ Mathematical, Physical and Life Sciences → MPLS
- ✓ Planning and Resource Allocation Committee → PRAC
- ✓ Pro-Vice-Chancellor → PVC
- ✓ Portable Document Format → PDF

When using an acronym that may be unfamiliar to your readers, spell it out in full the first time it is mentioned, with the acronym following in brackets; thereafter, use the acronym alone.

- ✓ The decision was made by the Planning and Resource Allocation Committee (PRAC). There are several meetings of PRAC every term.

Specific abbreviations

ampersands

Ampersands should only be used if they are part of official titles or names. Otherwise, spell out 'and'.

- ✓ Johnson & Johnson
- ✓ Uehiro Foundation on Ethics and Education

people's initials

Use a space to separate each initial.

- ✓ J R R Tolkien
- ✓ C S Lewis

measurements

When discussing large numbers in text, it is fine to use k/m/bn as shorter ways of spelling out 1,000/1,000,000/1,000,000,000 (or writing out 'one thousand'/'one million'/'one billion'), as long as you are consistent throughout the document. For multiple millions/billions you can use a mixture of words and numbers (eg 7 million, 8bn); again, ensure you are consistent throughout.

names of universities, degrees etc

See **Names and titles** for details.

Latin abbreviations

If you are using Latin abbreviations, make sure you know what they mean and when to use them. Do not use full stops after them and don't italicise them – see the **Highlighting/emphasising text** section for when to italicise.

etc [*et cetera*] – means 'and the rest'; use to indicate the continuation of a list

- ✓ Oxford offers many language courses: Russian, French, Spanish etc [the list could continue with the other language courses offered].

eg [*exempli gratia*] – means 'for example' or 'such as'; use with examples which are not exhaustive (and do not follow with a comma)

- ✓ Oxford offers many language courses, eg Russian, French, Spanish [those are some, but not all, of the language courses offered].

ie [*id est*] – means 'that is'; use with definitions or lists which are exhaustive (and do not follow with a comma)

- ✓ Catch a Blackbird Leys bus, ie numbers 1, 5 or 12 [those are the only buses which go to Blackbird Leys].

ibid [*ibidem*] – means 'in the same place'; use when making a subsequent reference/citation to a publication or other source mentioned in the immediately preceding note (ie no references to anything else have appeared in between)

- ✓ For a fuller explanation of telepathy, see Brown [*Speaking with the Mind*, Chicago (1945) p125]; Brown also gives further information on cats and telepathy [*ibid*, p229].

Capitalisation

General rule

Do **not** use a capital letter unless it is absolutely required.

Specific words

academic terms at Oxford

Capitalise the name but not the word 'term'.

- ✓ The Michaelmas term begins in October.
- ✓ The coldest part of the year usually falls in Hilary term.
- ✓ Finals take place in Trinity term.

If abbreviating term names, use MT, HT and TT.

- ✓ The post is vacant from MT 2014 until TT 2015.

Chancellor

Always capitalise when referring to the Chancellor of the University.

- ✓ Chris Patten is the Chancellor of the University.
- ✓ The University has had 192 Chancellors since 1224.

college

Capitalise only when used as part of the title of a college, not when referring to an institution without using its full name.

- ✓ Exeter College was founded in 1314. The college is one of the oldest in Oxford.
- ✗ Exeter College was founded in 1314. The College is one of the oldest in Oxford.

collegiate University

Capitalise 'University' but not 'collegiate'.

- ✓ We are seeking opinions from all members of the collegiate University.

department

Capitalise only when used as part of the title of a department, not when referring to a department without using its full name.

- ✓ The Department of Computer Science was previously known as the Oxford University Computing Laboratory. Both undergraduates and postgraduates study in this department.
- ✓ The Department for Work and Pensions has to make significant cuts this year, as do many government departments.

division

Capitalise only when used as part of the title of a division, not when referring to a division without using its full name.

- ✓ There are four academic divisions of the University: Humanities, Mathematical, Physical and Life Sciences, Medical Sciences and Social Sciences.
- ✓ The Medical Sciences Division is based mainly in Headington. The division's head is Alastair Buchan.

faculty

Capitalise only when used as part of the title of a faculty, not when referring to a faculty without using its full name.

- ✓ The Faculty of English is based in Manor Road. The faculty's phone number is 271055.

fellow

Capitalise only when used as part of an academic's formal title, not when referring to fellows in general.

- ✓ There are ten Fifty-Pound Fellows at All Souls.
- ✓ At its foundation, provision was made at All Souls for 40 fellows.
- ✗ At its foundation, provision was made at All Souls for 40 Fellows.

[Return to Contents](#)

4

government

Never capitalise, whether referring to a specific country's government or the concept of a government.

- ✓ The current British government is a coalition.

names with prefixes

Follow the preference of the individual, if known; if not, use lower case for the prefix. Alphabetise by the prefix.

- ✓ Dick Van Dyke is a star of daytime TV.
- ✓ Jan van Eyck painted in the 15th century.

professor

Capitalise only when used as part of an academic's formal title, not when referring to professors in general.

- ✓ It is common for Oxford professors to publish their works in learned journals.
- ✓ The Omega Solution is the latest contribution to research in the field by Professor Stephanie Archibald.

Reverend

Capitalise both 'Reverend' and 'The' (as well as other parts of the title).

- ✓ My tutor is The Reverend John Smith/The Very Reverend John Smith.

small caps

Do not use small caps, even for BC and AD.

- ✓ Egypt's Old Kingdom period began c2700 BC.

tutor

Capitalise only when used as part of an academic's formal title, not when referring to tutors in general.

- ✓ The Oxford tutorial system creates strong ties between students and their tutors.
- ✓ Dr Obadiah Braithwaite is the Tutor in Embroidery at Magdalen.

university

Capitalise only when used as part of the title of a university or when referring to the University of Oxford (both when 'University' is used as a noun and when it is used as an adjective).

- ✓ Oxford University is a large employer. The University has ~10,000 staff members.
- ✓ The University has four academic divisions.
- ✓ The event is open to all members of the collegiate University.
- ✓ The largest University division is Medical Sciences.
- ✓ Funding for universities has been cut recently.
- ✓ She attended the University of Liverpool to study English. It's a well-respected university and course.

Titles

People

See **Names and titles** for details.

Books/films/songs/games etc

Capitalise the first word of the title, and all words within the title except articles (a/an/the), prepositions (to/on/for etc) and conjunctions (but/and/or etc). See **Highlighting/emphasising text** for details on italicising and **Punctuation** for quotation mark advice.

- ✓ *The Last Mohican*
- ✓ *Far from the Madding Crowd*
- ✓ *Gone with the Wind*
- ✓ *World of Warcraft*
- ✓ *Grand Theft Auto V*
- ✓ 'Always Look on the Bright Side of Life'
- ✗ 'Always Look On The Bright Side Of Life'

[Return to Contents](#)

5

Subtitles

Capitalise subtitles only if the original title is printed in that way.

- ✓ *The Tale of Samuel Whiskers, or The Roly-Poly Pudding*
- ✓ *Dr Strangelove or: How I Learned to Stop Worrying and Love the Bomb*

Headlines, journal articles, chapter titles and lecture titles

Only capitalise the first word, any proper nouns and the first word following a full stop/question mark/exclamation mark.

- ✓ 'Who speaks for climate? Making sense of media reporting on climate change'
- ✓ Rock rafts could be 'cradle of life'
- ✓ 'Multiplicity of data in trial reports and the reliability of meta-analyses: empirical study'

Webpages

See **Miscellaneous** for advice on capitalisation of URLs, email addresses etc.

Numbers

How to write numbers

Spell out whole-number words for one to ten; use figures for numbers above ten.

- ✓ There were two people in the queue ahead of me, and six behind me.
- ✓ I need to buy Christmas presents for 12 people this year.

Use a combination of a figure and a word for very large round numbers (such as multiple millions/billions etc), or abbreviate it to 'm', 'bn' etc.

- ✓ The population of the earth is now 7 billion people.
- ✓ The population of the earth is now 7bn people.
- ✓ The budget came in at just under £2m.

If there are a lot of figures in a paragraph or text, some above ten and some below, use figures throughout to allow easy comparison by readers.

- ✓ There were 2 people in the queue ahead of me, and 22 behind me. The queues for other advisors had 10, 3 and 12 people.

Spell out words for 'first', 'second' and so on up to and including 'tenth'; use numbers and 'st'/'nd'/'rd'/'th' for larger ordinal numbers. Don't use superscript (to prevent problems with line spacing).

- ✓ She was the first person from her school to get a place at Oxford.
- ✓ He got an upper second, to his relief.
- ✗ She got a 3rd class degree.
- ✓ The 17th president of the United States was Andrew Johnson.

Always use figures and symbols for percentages, measurements and currency. Use commas to punctuate large numbers.

- ✓ Question 12 is worth 10% of the available marks.
- ✗ 20 per cent of commuters use their cars.
- ✓ The average height of a woman in the UK is 1.61m.
- ✓ The cost, at £5.99, was less than their overall budget of £50.
- ✓ The population of New York City is estimated to be 8,008,278.

Times

Use either the 12- or 24-hour clock – not both in the same text. The 12-hour clock uses a full stop between the hours and minutes; the 24-hour clock uses a colon and omits am/pm.

- ✓ The lecture starts at 11.30am and ends at 1pm.
- ✓ The lecture starts at 11:30 and ends at 13:00.
- ✗ The lecture starts at 11.30am and ends at 13:00.
- ✗ The lecture starts at 16:00pm.

Use 'noon' or 'midnight' instead of '12', '12 noon' or '12 midnight'.

- ✓ The closing date for applications is noon on 12 July.

If using the 12-hour clock, don't use additional '.00' for times on the hour, and close up space between the number and the 'am' or 'pm'.

- ✓ The lecture starts at 9am.
- ✓ The lecture starts at 11.30am and ends at 1pm.
- ✗ The lecture starts at 9.00am.
- ✗ The lecture starts at 9 am.

Dates

Always put the date before the month.

✓ Easter this year is on 13 April.

✗ Easter this year is on April 13.

Don't use 'th' etc with dates – just the number and month – and never precede the number with 'the'.

✓ Easter this year is on 13 April.

✗ 11th November is Armistice Day.

✗ Armistice Day is on the 11 November.

Use days with dates only for emphasis or the avoidance of confusion/ambiguity.

✓ The wedding is on 30 December.

✗ The wedding is on Saturday 20 December.

✓ The Modern Superstitions conference is on Friday 13 April.

Spans of numbers and years

Shorten periods where it is not ambiguous to do so and use the shortest text possible. However, do not elide numbers between 11 and 19, which must always be written in full (as they would be spoken).

✓ The 'short twentieth century' refers to the period 1914–91.

✓ The First World War (1914–18) was shorter than the Second World War (1939–45).

✗ The First World war lasted from 1914–8.

✓ The professorship was held 1993–5 by Alice Jenkins.

✗ Inner-city flats cost £100–£200,000. [Price could start at £100 or £100,000.]

To refer to an academic or financial year, you can use either the format '2011–12' or '2011/12' – but ensure you are consistent throughout the text.

✓ The Proctors for 2013–14 will be elected in the 2011–12 academic year.

✓ Profits are up year on year: the company did better in 2011/12 than in 2010/11.

If using 'from' with a start date/time, always use 'to' to indicate the end date/time rather than an n–dash; alternatively, just use an n–dash without 'from'.

✓ Michaelmas term runs from October to December.

✓ Michaelmas term runs October–December.

✗ Michaelmas term runs from October–December.

Punctuation

General rule

Use as little punctuation as necessary while retaining the meaning of the sentence.

Apostrophe

to indicate possession

Use 's after singular nouns, plural nouns which do not end in s and indefinite pronouns.

✓ Frank's book

✓ anybody's guess

✓ The children's play area is next to the men's toilet.

Use just ' after plural nouns ending in s.

✓ Strong tea is sometimes called builders' tea.

If a name already ends in s or z and would be difficult to pronounce if 's were added to the end, consider rearranging the sentence to avoid the difficulty.

✓ Jesus's methods were unpopular with the ruling classes. OR
The methods of Jesus were unpopular with the ruling classes.

In compound nouns and where multiple nouns are linked to make one concept, place the apostrophe at the end of the final part (and match it to that noun).

✓ the Archbishop of Canterbury's tortoise

✓ my mother-in-law's dog

✓ his step-brothers' cars

✓ Lee and Herring's *Fist of Fun*

Do not use an apostrophe in **its** with the meaning 'belonging to it' (this is analogous with his/hers/theirs): note that **it's** is a contraction of 'it is'.

✓ The cat has been out in the rain and its paws are muddy.

✓ The cat has been out in the rain and it's muddy.

✗ The cat has been out in the rain and it's tail is wet.

Some place names have an apostrophe and some don't – this can't be predicted and must be checked.

✓ All Souls College

✓ Earls Court

✓ St Peter's College

✓ Land's End

✓ University of St Andrews

Some street names have an apostrophe (usually linked to saints' names from nearby churches); these are also idiosyncratic.

✓ There is a famous pub on St Giles'.

✗ St Giles's splits into Woodstock and Banbury Roads.

✓ Christ Church is on St Aldate's.

✓ St Michael's Street is a through road for bicycles.

Use apostrophes with noun phrases denoting periods of time (use an apostrophe if you can replace the apostrophe with 'of').

✓ He took a week's holiday [holiday of a week].

✓ You must give three months' notice [notice of three months].

But do not use an apostrophe in adjectival phrases.

✓ She was eight months pregnant when she went into labour.

Apostrophe (cont)

to indicate that letters have been omitted (contractions)

Use an apostrophe in the position the omitted letters would have occupied, not where the space was between the original words.

✓ I don't like cheese. [=do not]

✗ I do'n't like cheese. [=do not]

✓ He wouldn't do that.

Do not use an apostrophe before contractions accepted as words in their own right.

✓ He is on the phone.

✓ He had swine flu.

✗ There is no vaccine for all types of 'flu.

Do not use an apostrophe to make a plural, even with a word/phrase that is not usually written in the plural or which appears clunky. All of the following examples take an **s** as normal in English to make their plurals.

✗ Three video's for a tenner.

✗ I trust all the MP's.

✗ Clothes were colourful in the 1970's.

✗ CD's will soon be obsolete.

✗ This is a list of do's and don't's.

To clarify something which will look odd if an **s** is added, consider italicising it or placing it in single quotation marks.

✓ Subtract all the *x*s from the *y*s.

✓ Dot the 'i's and cross the 't's.

Brackets

round brackets ()

Use in place of a pair of dashes or commas around a non-defining phrase (one which adds extra information, a translation, dates, an explanation or a definition).

✓ The library (which was built in the seventeenth century) needs to be repaired.

✓ It was (as far as I could tell) the only example of its kind.

✓ Magdalen College (founded in 1458) has a herd of deer.

✓ The tactic of *Blitzkrieg* (which means 'lightning war' in German) was used in the invasion of Poland in 1939.

✓ Preheat the oven to 350°F (180°C).

using other punctuation with brackets

Include full stops/exclamation marks/question marks/quotation marks before the closing bracket only if the complete sentence/quote is in brackets; otherwise, punctuate after the closing bracket.

✓ The last bus today is at 4.45 (which is earlier than usual).

✓ The last bus today is at 4.45. (That's earlier than usual.)

square brackets []

Use to enclose comments, corrections, references or translations made by a subsequent author or editor.

✓ An article referring to the restrictions placed by some airlines on the appearance of female cabin crew stated that even footwear was proscribed [sic].

✓ I have been responsible in the real sense, that I have had the blame for everything that has gone wrong. [Laughter and cheers.]

✓ This was quoted by Brown [1940, Chicago].

angle brackets < > and curly brackets { }

These are used for technical purposes – only use them in the correct context.

Bullet points

Don't punctuate the end of bullet points which are a list of items.

✓ 2014 concert performers:

- Slade
- The Smiths
- Metallica
- the Spice Girls

If the bullet points form a complete sentence with preceding text, add a full stop to the end of the last point.

✓ We are holding a concert in 2014, at which the following acts will perform:

- Slade
- The Smiths
- Metallica
- the Spice Girls.

If text inside the bullet point is a complete sentence in its own right, add a semicolon to the end of each point, 'or' or 'and' (depending on the sense of your sentence) to the end of the penultimate point, and a full stop to the end of the last one.

- ✓ The following will be considered good reasons for missing the final meeting of the year:
 - there was a postal strike. This only applies if the postal strike took place before the date of the meeting and if you have not signed up for email alerts;
 - you are absent as a result of illness;
 - you are unable to attend because of problems with public transport (proof of this will be required);
 - there is something more interesting happening elsewhere which you would rather attend; or
 - you have obtained a ticket to see the Spice Girls in concert.

Colon and semicolon

Use a colon to introduce a subclause which follows logically from the text before it, is not a new concept and depends logically on the preceding main clause.

✓ When I was young, I went on two holidays: to the Lake District and to Cornwall.

✓ A new drink was introduced to Britain: tea.

Do not use a colon if the two parts of the sentence are not logically connected.

✗ I used to be slim: I will try to lose weight.

✓ I would like to be slim: I will try to lose weight.

✗ We were in trouble this time: we'd never been in trouble before.

✓ We were in trouble this time: the lid had come right off.

✓ There are two parts to this sentence: the first part, which precedes the colon, and the second part, which doesn't.

Use a semicolon to link two related parts of a sentence, neither of which depends logically on the other and each of which could stand alone as a grammatically complete sentence.

✓ The best job is the one you enjoy; the worst job is the one you hate.

✓ It is a far, far better thing that I do, than I have ever done; it is a far, far better rest that I go to, than I have ever known.

Use semicolons in place of commas in a complicated list or sentence if it will improve clarity, particularly if list items already include commas.

✓ We plan to review the quality of the research of the department, including its participation in interdepartmental, interdivisional and interdisciplinary activities; its research profile and strategy; and future challenges and opportunities.

✓ I visited the Ashmolean Museum, Oxford; the Victoria and Albert Museum, London; and the Pencil Museum, Keswick.

Comma

Use a pair of commas to surround a non-defining clause (one which adds descriptive information but which can be removed without losing the meaning of the sentence) – note that only 'which' or 'who' can be used in this type of clause, not 'that'.

✓ The library, which was built in the seventeenth century, needs to be repaired.

✓ The man, who climbed the tower without a safety harness, died of old age.

Do not use commas to surround a defining clause (which cannot be removed without losing the meaning of the sentence) – note that 'which' or 'who' can be replaced by 'that' in this type of clause.

✓ The library which was built in the seventeenth century needs to be repaired [but the library which was built in the eighteenth century does not].

✓ The man that climbed the tower without a safety harness died of old age [but the other man died in a different way].

✓ He asked his friend Sam to be his second [not any of his other friends].

Use commas to surround a non-defining word or phrase (which adds information but could be omitted without changing the sense of the sentence), and follow the non-defining word/phrase with a single comma if it is at the start of the sentence.

✓ Shakespeare, the prolific playwright, might not have existed.

✓ A prolific playwright, Shakespeare might not have existed.

✓ He asked Sam, his friend, to be his second [not the Sam who is his barber].

✓ The prime minister, David Cameron, is an alumnus of Brasenose.

Do not use a comma where defining information is used at the start of a sentence.

✓ The prolific playwright Shakespeare might not have existed.

✗ The prolific playwright, Shakespeare might not have existed.

✓ His friend Sam was his second.

✗ His friend, Sam was his second.

Defining vs non-defining information

Do not use a comma to join two main clauses, or those linked by adverbs or adverbial phrases (eg 'nevertheless', 'therefore', 'however'). This is sometimes referred to as 'comma splicing'. Either use a semicolon or add a coordinating conjunction (eg 'and', 'but', 'so').

✓ Shakespeare was popular, and his plays were all profitable.

✓ Shakespeare was popular; his plays were all profitable.

✗ Shakespeare was popular, his plays were all profitable.

Use a comma after an introductory adverb, adverbial phrase or subordinate clause; or use a pair of commas surrounding it if it is in the middle of a sentence.

✓ However, it was too late for that.

✓ It was, however, too late for that.

✓ With his possessions in a bundle, Dick Whittington walked to London.

✓ Dick Whittington, with his possessions in a bundle, walked to London.

Do not use a comma after a time-based adverbial phrase.

✓ After playing tennis all day she was tired.

✓ Whenever she went to the cinema she ate popcorn.

✓ In 2010 the most popular game among children was hopscotch.

Use a comma between multiple qualitative adjectives (those which can be used in the comparative/superlative or modified with 'very', 'quite' etc).

✓ He was a big, fat, sweaty man with soft, wet hands.

Do not use a comma between multiple classifying adjectives: absolutes which either are or are not, such as 'unique', 'English', 'black' etc (although note that stylistically these can be modified).

✓ It was an edible German mushroom.

✓ The eighteenth-century sandstone tower is lit up at night.

Do not use a comma between classifying and qualitative adjectives.

✓ It was a large German mushroom with hard black edges.

✓ It was a large, squishy German mushroom with hard, frilly black edges.

Use a comma between items in a list.

✓ I ate fish, bread, ice cream and spaghetti.

✓ I have nothing to offer but blood, toil, tears and sweat.

Note that there is no comma between the penultimate item in a list and 'and'/'or', unless required to prevent ambiguity – this is sometimes referred to as the 'Oxford comma'. However, always insert a comma in this position if it would help prevent confusion.

✗ He took French, Spanish, and Maths A-levels.

✓ I ate fish and chips, bread and jam, and ice cream.

✓ We studied George III, William and Mary, and Henry VIII.

✗ She left her money to her parents, Mother Theresa and the pope.

Dashes and hyphens — – -

m-dash (—)

Do not use; use an n-dash instead.

n-dash (–)

Use in a pair in place of round brackets or commas, surrounded by spaces.

✓ It was – as far as I could tell – the only example of its kind.

✓ The library – which was built in the seventeenth century – needs to be repaired.

Use singly and surrounded by spaces to link two parts of a sentence, in place of a colon.

✓ The bus was late today – we nearly missed the lecture.

Use to link concepts or ranges of numbers, with no spaces either side.

✓ German–Polish non-aggression pact

✓ The salary for the post is £25,000–£30,000.

✓ Radio 1 is aimed at the 18–25 age bracket.

Use between names of joint authors/creators/performers etc to distinguish from hyphenated names of a single person.

✓ Lennon–McCartney compositions

✓ Superman–Batman crossover comics

hyphen (-)

When to use a hyphen

In an adjectival phrase before a noun

- ✓ the up-to-date list
- ✓ The value of a first-class degree is indisputable.
- ✓ a hot-air balloon
- ✓ 'Rethinking provincialism in mid-nineteenth-century narrative fiction: Villette from our village'

In an adjectival phrase including a verb participle

- ✓ The jumper was tight-fitting.

With prefixes only if required to avoid confusion/mispronunciation, such as where prefixes themselves or letters are repeated

- ✓ predynastic Egypt
- ✓ gifts of pre-eminent objects and works of art to the nation
- ✓ The animals are re-released into the wild when recovered.
- ✓ A protein precursor can also be called a pro-protein.
- ✓ Procapitalists and anticapitalists clashed in the streets.
- ✓ The email address for the webmaster can be found on the website.

With prefixes before a proper name, number or date

- ✓ anti-Thatcherism
- ✓ pre-2000 politics
- ✓ Hilary term starts in mid-January.

In numbers which are spelt out

- ✓ Twenty-seven is the most popular 'random' number.
- ✓ The Thirty-Nine Steps

In compass points (unless used geographically rather than as directions)

- ✓ They're heading south-east.
- ✓ nor'-nor'-east
- ✓ The southwest is a popular holiday destination.

When not to use a hyphen

In noun phrases

- ✓ Labour Party conference
- ✓ The 19th century saw much reform.

To make a new compound noun – if it is a recognisable concept, make it one word; if it isn't, use two words

- ✓ Websites are made up of webpages.
- ✓ Send me an email when you're ready to proceed.
- ✗ Send me an e-mail.

In an adjectival phrase following a noun

- ✓ The list was up to date.
- ✓ His marks just scraped into the first class.
- ✗ She wasn't top-drawer.

In an adjectival phrase before a noun where the first element is an adverb ending in -ly (but note that any other adverbs in adjectival phrases do take a hyphen)

- ✓ She had a finely tuned ear for off-key music.
- ✓ XML documents must be well-formed texts.
- ✗ She was a highly-respected tutor.
- ✓ She was a badly paid apprentice.

Ellipsis...

Use an ellipsis to show that some text is missing, usually from a quotation – do not surround it with spaces.

- ✓ ...we shall fight on the beaches...we shall never surrender...
- ✓ It is a truth universally acknowledged...

There is no need to add square brackets around an ellipsis.

- ✗ [...]we shall fight on the beaches[...]

Use an ellipsis to indicate a pause for comic or other effect – follow the ellipsis with a space in this case, as it stands in place of a comma or full stop.

- ✓ You don't have to be mad to work here... but it helps!

Note that, if used either in place of omitted text at the end of a clause/sentence or to indicate a pause for effect, a full stop/comma should not follow the ellipsis. However, an exclamation mark or a question mark can and should follow the ellipsis if required.

- ✓ Are you...?
- ✓ Did he say that...?

Use an ellipsis to indicate a trailing off in speech or thought.

- ✓ We could do this...or maybe that...

Full stop, exclamation mark and question mark

Use one – but only one – of these at the end of every sentence.

- ✓ What time did you leave last night?
- ✓ We went home at 5 o'clock.
- ✓ Go home now!

Do not use a full stop at the end of titles, even if they make a sentence, but, if a title ends with an exclamation mark or question mark, do include it.

- ✓ *All's Well that Ends Well* is my favourite play.
- ✓ 'Will You Still Love Me Tomorrow?' was a hit for the Shirelles.
- ✓ 'Help!' was covered by Bananarama in 1989.

Do not use a full stop if it will be followed, or preceded, by an ellipsis.

- ✗ Behind him stood a figure. ...It was ghostly grey.

Use a full stop, not a question mark, at the end of a reported question – only use a question mark for a direct question (whether in quotation marks or not).

- ✓ He asked if I wanted to go home that morning.
- ✓ 'Do you want to go home this morning?' he asked.
- ✗ He asked if I wanted to go home?

Use a full stop, not an exclamation mark, at the end of a reported imperative.

- ✓ Wait for me! → He asked me to wait for him.

Quotation marks

Use single quotation marks for direct speech or a quote, and double quotation marks for direct speech or a quote within that.

- ✓ 'I have never been to Norway,' he said, 'but I have heard it described as "the Wales of the North".'

Use no quotation marks if the quote is displayed (ie not in line with the rest of the text).

- ✓ as I noted then,
Those of us who toil in the Groves of Academe
know full well that our research helps inform
our teaching...

Use single quotation marks and roman (not italic) type for titles that are not whole publications: eg short poems, short stories, songs, chapters in books, articles in periodicals etc. See also **Highlighting/emphasising text**.

- ✓ I, Robot contains nine short stories, of which 'Little Lost Robot' is my favourite.
- ✓ Queen's 'Bohemian Rhapsody', from the album *Night at the Opera*, reached number one in both 1975 and 1991.

Using other punctuation with quotation marks

If the quote would have required punctuation in its original form, place the punctuation inside the quotation marks. (If it is unclear, try writing the whole sentence out without quotation marks and 'he said' etc, and replicate the resulting punctuation.)

- ✓ Bob likes cheese. → 'Bob', I said, 'likes cheese.' OR 'Bob likes cheese,' I said.
- ✓ Bob, do you like cheese? → 'Bob,' I asked, 'do you like cheese?'
- ✓ Out, damn'd spot! → 'Out,' said Lady Macbeth, 'damn'd spot!'
- ✓ 'You're engaged to Florence?' I yipped, looking at him with a wild surmise.

Place any punctuation which does not belong to the quote outside the quotation marks (except closing punctuation if the end of the quote is also the end of the sentence).

- ✓ After all, tomorrow is another day. → 'After all,' said Scarlett, 'tomorrow is another day.' OR 'After all, tomorrow', said Scarlett, 'is another day.'
- ✗ 'The kitchen,' he said, 'is the heart of the home'.
- ✓ 'The kitchen', he said, 'is the heart of the home.'

Note that American English has different rules about the use of quotation marks.

Names and titles

General titles

Use capitals for titles prefixing names, but not for job descriptions. Note that some job descriptions are never used with names, such as 'prime minister'.

- ✓ Although being president of the United States is stressful, President Obama was glad to be re-elected.
- ✓ The prime minister of Great Britain and Northern Ireland is the leader of the party that wins the most seats. The Right Honourable David Cameron MP is the current prime minister.
- ✓ The current pope, Pope Francis, is Argentine.

Give people's title, forename and surname when first mentioned. On subsequent mentions, use either surname only or title and surname (unless further information is required to prevent ambiguity), but be consistent with whichever usage you choose.

- ✓ Dr John Smith was present at the ceremony, as was Professor Susan Jones. Dr Smith had to leave early.
- ✓ Dr John Smith and Professor Susan Jones presented their research paper to a large audience. The results will be published in book form, which Smith says will be available in the spring.
- ✗ Dr John Smith and Professor Susan Jones debated the topic. Smith recently reviewed Professor Jones's book.

Note that it can be helpful to your readers to clarify the sex of the person if it is unclear (eg if they have a name given to men and women, or an unusual name).

Oxford-specific titles

Use capitals when referring to the specific person holding a specific position and to their work in this role, but not when referring to any holder of that role unless it is a statutory position (see **Capitalisation** and **Word Usage** sections for further information).

- ✓ Andrew Hamilton became Vice-Chancellor in 2009.
- ✓ There are several Pro-Vice-Chancellors without portfolio.
- ✓ The Registrar will always have to attend these meetings.
- ✓ He invited Wadham's Head of House, Lord Macdonald, to attend the event. Other heads of house were not invited.
- ✓ I wonder who the Senior Proctor will be next year...
- ✓ Candidates will be required to undertake practical work, as specified by the Head of the Department of Experimental Psychology.
- ✓ Recruiting new academic staff is vital to all departments; heads of department often personally oversee the procedure.

Other titles

members of the peerage

When referring to or writing to people entitled to call themselves Sir/Dame/Lord/Lady etc, make sure that you know the correct form of address for that individual. The examples below are not definitive – if in doubt, consult Debrett's (for general advice and examples: www.debretts.com/forms-address/titles) or Who's Who (for specific individuals: www.ukwhoswho.com).

If an individual has expressed a wish to be addressed in a particular way, even if it is technically incorrect, use their preferred style.

Take particular care with people from countries where family names precede given names.

For more advice on addressing people with titles, see the helpful guide compiled by the University Development Office at: www.advancingoxford.ox.ac.uk/file/Correct-Form-Booklet-for-Onscreen-Viewing.pdf.

knights/dames

Always use first names with these titles, whether or not you are using surnames as well.

- ✓ 'Are you going to hear Sir John Smith's speech? Sir John is always a good public speaker.'
- ✓ Dame Jane Jones is the chair of this committee.

If you are writing to a knight or a dame, use 'title first name surname' on envelopes then just 'title first name' in the salutation.

- ✓ To: Dame Jane Jones, 14 Bluebird Way, Oxford OX1 1AB
- ✓ Dear Sir John...

lords/ladies

Check the exact status of anyone verbally addressed as Lord X or Lady Y as these titles may be used by many types of peer (eg earls, barons, viscounts, sons of dukes etc) whose titles in writing are different.

Life peers are formally barons/baronesses but are addressed informally as Lord/Lady followed by the name they chose when ennobled.

- ✓ Helena Kennedy (Baroness Kennedy of the Shaws) should be addressed as Lady Kennedy of the Shaws.
- ✓ Baron Patten of Barnes is Chancellor of the University. Lord Patten was formerly Chairman of the BBC Trust.
- ✗ Have you met Lord Chris Patten?
- ✓ Lord Sugar's full title is Baron Sugar of Clapham.
- ✓ Lady Benjamin was the best *Playschool* presenter.

promotion within an order of chivalry

If someone is promoted within an order of chivalry (eg from MBE to OBE), the higher honour replaces the lower; don't list all of them.

- ✓ Mrs Tanni Grey-Thompson was appointed MBE in 1993.
- ✓ Mrs Tanni Grey-Thompson was appointed OBE in 2000.
- ✓ Dame Tanni Grey-Thompson was appointed DBE in 2005.

If someone receives an honour in a different order of chivalry, or is made a life peer, they are entitled to use both honours but not both titles.

- ✗ Dame Tanni Grey-Thompson, OBE MBE DBE, is a fellow of Hertford.
- ✓ Baroness Grey-Thompson, DBE, was made a life peer in 2010.
- ✗ Baroness Dame Grey-Thompson

Combining titles

For someone entitled to a styling such as Sir/Dame as well as an academic title, the academic title always precedes the Sir/Dame if both are used.

- ✓ Professor Sir John Smith
- ✓ Professor Dame Susan Jones
- ✗ Sir Professor John Smith

For Reverends, Right Reverends, Very Reverends etc, the religious title precedes the academic title. Always use a capitalised 'The' before the title.

- ✓ The Revd Dr Giles Fraser resigned from St Paul's.
- ✓ The Revd Professor Andrew Linzey is a member of the Faculty of Theology.
- ✗ Dr Revd Giles Fraser spoke to the press about his decision to resign.

If someone who has been ordained in the clergy of the Church of England is subsequently honoured, they do not use Sir/Dame as a title but can use the appropriate postnominal letters.

- ✓ Professor Diarmaid MacCulloch, Kt, is a Deacon of the Church of England.

In a shortened subsequent mention, use either Sir/Dame/The Revd or the academic title, not both – consider which title is more appropriate for the context (academic titles are often better for University purposes). Note that non-academic titles should not be used with surnames only and that 'The Revd' should not be used with surnames alone; revert to Dr/Mr/Miss/Mrs as appropriate.

- ✓ Sir John
- ✓ Professor Jones
- ✓ Dr Fraser
- ✗ The Revd Fraser
- ✗ Revd Professor Linzey
- ✓ Professor Sir John Baggins will attend the lecture to be given by The Revd Dr Peter Precise in the Examination Schools on Thursday. Dr Precise and Sir John are both senior members of the Oxford University Tiddlywinks Club, and Dr Precise is the author of the bestselling *Pedantry and Hypothetical Names*. He is not to be confused with his colleague, The Revd Bona-Exempla Familyname. Miss Familyname will give her lecture next week.

Postnominals

Remember that you do not need to list all awards, degrees, memberships etc held by an individual – only those items relevant to your writing.

The order for postnominals is:

- civil honours
- military honours
- QC
- degrees, in the order
 - bachelors'
 - masters'
 - doctorates
 - postdoctoral
- diplomas
- certificates
- membership of academic or professional bodies.

Do not include a BA for Oxford/Cambridge graduates if they also have the MA.

academic qualifications

In order of academic precedence; masters' degrees with * are equal to an MA for this purpose

- **DD** (Doctor of Divinity)
- **DCL** (Doctor of Civil Law)
- **DM** (Doctor of Medicine)
- **DLitt** (Doctor of Letters)
- **DSc** (Doctor of Science)
- **DMus** (Doctor of Music)
- **DPhil** (Doctor of Philosophy)
- **DClinPsych** (Doctor of Clinical Psychology)
- **DEng** (Doctor of Engineering)
- **MCh** (Master of Surgery)
- **MSc** (Master of Science)
- **MLitt** (Master of Letters)
- **MPhil** (Master of Philosophy)
- **MSt** (Master of Studies)
- **MTh** (Master of Theology)
- **MBA** (Master of Business Administration)

- **MFA** (Master of Fine Art)
- **MPP** (Master of Public Policy)
- **MA** (Master of Arts)
- **MBioChem** (Master of Biochemistry)*
- **MChem** (Master of Chemistry)*
- **MCompSci** (Master of Computer Science)*
- **MCompPhil** (Master of Computer Science and Philosophy)*
- **MEarthSci** (Master of Earth Sciences)*
- **MEng** (Master of Engineering)*
- **MMath** (Master of Mathematics or Mathematics and Statistics)*
- **MMathCompSci** (Master of Mathematics and Computer Science)*
- **MMathPhil** (Master of Mathematics and Philosophy)*
- **MPhys** (Master of Physics)*
- **MPhysPhil** (Master of Physics and Philosophy)*
- **BD** (Bachelor of Divinity)
- **BCL** (Bachelor of Civil Law)
- **MJur** (Magister Juris)
- **BM BCh** (Bachelor of Medicine and Surgery)
- **BLitt** (Bachelor of Letters)
- **BSc** (Bachelor of Science)
- **BMus** (Bachelor of Music)
- **BPhil** (Bachelor of Philosophy)
- **BA** (Bachelor of Arts)
- **BFA** (Bachelor of Fine Art)
- **BTh** (Bachelor of Theology)

Consider giving the name of the awarding university (using a shortened form if required and if easily recognisable (Oxf, Camb, UCL, MIT etc) if academic qualifications are relevant.

A space is used to separate degrees from the same institution, and a comma is used to separate sets of degrees from different institutions; if the same level degree has been awarded by more than one institution, list them in alphabetical order of institution.

✓ Professor Xavier Postlethwaite, QC, BEng PhD UCL, MA PhD Camb, MA DPhil Oxf, PhD Manc, FRS

✓ Sir Charles Overlord, VC, BA S'ton

See www.ox.ac.uk/gazette/calendar for further detail if required.

[Return to Contents](#)

20

Highlighting/emphasising text

Bold

Use bold sparingly to emphasise the part of your text you wish to stand out. This could be someone's name, a deadline date or another key piece of information.

Punctuation which follows bold text should not itself be bold (unless the whole sentence is in bold type).

If you are transcribing a speech, you might want to use bold text (or italic text) to emphasise words in writing on which the speaker placed particular emphasis in speech.

✓ The Man Booker Prize for 2012 was awarded to **Hilary Mantel**; this was the second time that she has won.

✓ Applications can either be made online or emailed to sample@email.com. The deadline for submissions is **noon on 2 April** and any applications received later than this will not be considered.

Italic

Use italics to flag part of your text which is different from that surrounding it.

Titles of books, journals, plays, films, musical works etc should be given in italics if they are a complete published work; if you are referring to an individual short story, song, article etc within a larger publication, use single quotation marks (see also **Quotation marks** under **Punctuation**).

✓ *I, Robot* contains nine short stories, of which 'Little Lost Robot' is my favourite.

✓ The number-one single in the hit parade this week is 'Candy' by Robbie Williams, from the album *Take The Crown*.

If the title includes 'The' or 'A' as the start of the title, italicise that as well.

✓ *A Tale of Two Cities* has perhaps the most famous opening sentence in English literature.

Use italics for foreign words and phrases embedded within your text, including species and genera names in Latin.

✓ There are nine *ex officio* members of Council.

✓ When producing its annual report, the committee shall consider, *inter alia*, any relevant HEFCE evaluations.

✓ A seven-sisters rose bush (*Rosa multiflora*) can be either white or pink.

✓ Dante tells us that above the entrance to the Inferno is inscribed *Lasciate ogne speranza, voi ch'entrate*.

Plurals, past tenses or other grammatical changes to italicised titles or phrases should only be italicised up to the end of the title or phrase – do not italicise the **s**, **ed** etc – and punctuation should only be italicised if it is part of the title, quote etc. Note that it is not always easy to tell whether a full stop/comma is in italic.

✓ The remaining *Gazettes* of Michaelmas term are published on 22 and 29 November and 6 December.

✓ Changes to regulations need to be *Gazetted* at least eight weeks before they are due to take effect.

Underlining

Avoid using underlining for emphasis; this generally suggests hyperlinks, especially on webpages.

[Return to Contents](#)

21

Word usage and spelling

Common confusions in word usage

Confused words	How they differ	Example
among vs between	<ul style="list-style-type: none">Among is used for undifferentiated items.Between is used with individual, named items.	<ul style="list-style-type: none">She couldn't decide among all the colleges.She couldn't decide between Magdalen or St Hilda's.
mutual vs reciprocal	<ul style="list-style-type: none">Mutual is used when more than one person has the same feeling/opinion as another towards a third party/object/concept etc.Reciprocal is used when two or more people feel, think or act in the same way about or to one another.	<ul style="list-style-type: none">'I disagree with the government's policy on carrots.' 'So do I. The feeling's mutual.''I won't steal your cheese.' 'I won't steal your cheese either. We have a reciprocal arrangement.'
less vs fewer	<ul style="list-style-type: none">Less is used with nouns which are not countable objects: if you could use much to describe having a lot of the noun, use less.Fewer is used with countable objects: if you could use many to describe having a lot of the noun, use fewer.	<ul style="list-style-type: none">'I can't eat that much cheese: please give me less.''I can't eat that many sprouts: please give me fewer.'
effect vs affect (verb)	<ul style="list-style-type: none">Effect as a verb means to bring about, or to have the result that.Affect as a verb means to have an impact on or to change something; it also means to simulate something which is untrue.	<ul style="list-style-type: none">A glass of brandy may effect his recovery [bring his recovery about].A glass of brandy may affect his recovery [have an impact on whether he recovers].He affected to have drunk only one glass of brandy [when he had actually drunk more than one glass].
effect vs affect (noun)	<ul style="list-style-type: none">Effect as a noun means the impact something causes.Affect as a noun means someone's outward appearance of their psychological state.	<ul style="list-style-type: none">The storm had wide-reaching effects.His affect was one of cheerful indifference.
infer vs imply	<ul style="list-style-type: none">Infer is to read a meaning into a statement which has not been explicitly stated: to read between the lines.Imply is to suggest something without explicitly stating it: to hint at something (usually something negative).	<ul style="list-style-type: none">He told me that these one-size-fits-all gloves fit most people's hands. I inferred that he thought my hands were too big, and resented what he was implying.
compared to vs compared with	<ul style="list-style-type: none">Comparing something to another thing highlights a (perhaps metaphorical) similarityComparing something with another thing highlights the differences between them	<ul style="list-style-type: none">Shall I compare thee to a summer's day?Price comparison websites allow you to compare one company's prices and policies with those of their competitors.

[Return to Contents](#)

22

Spelling

General guidelines

Use suffix **-ise/-yse/-isation** not **-ize/-yze/-ization**. Note that this is different from OUP's choice, because OUP prefers to reflect the Greek roots of some such words. We prefer '-ise' as it is more common in British usage and requires fewer exceptions.

- ✓ Derren Brown hypnotised his subject live on TV.
- ✓ They didn't receive authorisation to take the trip.
- ✗ She wished she had organized her books alphabetically.

Retain **-e** where required for pronunciation: ageing/acknowledgement.

Contractions: use of 'hasn't' rather than 'has not' etc is fine in the majority of cases, especially informally.

Foreign spellings

Just use 'e' spellings, not *ae* or *æ*, where in common British usage.

- ✓ encyclopedia
- ✓ medieval

Technical words retain the ligature.

- ✓ archaeology
- ✓ hæmatology
- ✓ orthopædics

Use accents and different letters in foreign words (ø, ç, capitalisation for German nouns etc) only when:

- a word is still considered foreign and has not (yet) been absorbed into English
- they are required to differentiate from another word (in English or the source language)
- they are required as part of the name of a person, place, book etc.

Don't use accents on capital letters.

[Return to Contents](#)

Plurals

Use appropriate foreign (particularly ancient Greek and Latin) plural forms where still in common usage (also see **alumna/ae/i/us** below).

- ✓ nucleus → nuclei
- ✓ stratum → strata
- ✓ genus → genera
- ✓ analysis → analyses
- ✓ basis → bases
- ✓ crisis → crises
- ✓ phenomenon → phenomena
- ✓ bacterium → bacteria
- ✓ millennium → millennia

Note that more than one plural form is sometimes in use for different meanings of a word.

- ✓ formula → formulas **but** formulae in maths/chemistry
- ✓ index → indices for maths **but** indexes for books
- ✓ appendix → appendices for books **but** appendixes in anatomy
- ✓ medium → media **but** mediums for spiritualists
- ✓ datum → data in technical cases **but** points of data in everyday use

Proper names ending in -y do not change to -ies if pluralised.

- ✓ One of the most popular stories in *Bunty* was 'The Four Marys'.

With compound words formed by a noun and an adjective, or two nouns connected by a preposition, pluralise the (more important) noun.

- ✓ Attorney General → Attorneys General
- ✓ brother-in-law → brothers-in-law
- ✓ passer-by → passers-by
- ✓ gin and tonic → gins and tonic

[Return to Contents](#)

23

Tricky words

- accommodation
- alumna – female former member (of college etc)
- alumnae – plural form for female-only former members (of college etc)
- alumni – plural form for either male-only or mixed-gender former members (of college etc)
- alumnus – male former member (of college etc)
- benefited
- biased
- comprise (not comprise of)
- co-operation (hyphenated)
- dependant (noun)/dependent (adjective)
- email (lower case and no hyphen)
- enquire/enquiry (preferred to inquire/inquiry)
- focused
- fundraising
- instalment
- internet (lower case)
- judgement (moral, academic etc)
- judgment (legal decision only)
- liaison/liaison
- manoeuvre
- no-one (hyphenated)
- paralleled
- postdoctoral (no hyphen)
- postgraduate (no hyphen, whether as noun or adjective)
- postholder (no hyphen and lower case)
- principal (noun or adjective: chief, main, head)
- principle (noun only: ethical standpoint)
- program (computer applications only)
- programme (schedule of events)
- riveting
- stationary (not moving)
- stationery (paper, pens etc)
- supersede
- till (not 'til: not an abbreviation of until)
- website/webpage (no hyphen and lower case)

Words usually spelt differently in American English

These are given for information only – do not use the US spelling unless you are quoting an American speaker or from American text (in which case the original should be kept).

British	American	Examples
-our	-or	colour/color
-ise	-ize	organise/organize
-yse	-yze	analyse/analyze
-re	-er	centre/center
-ling	-ling	travelling/traveling
-lled	-led	travelled/traveled
-ller	-ler	traveller/traveler

The following words are spelt differently in British English depending on whether they are nouns or verbs, whereas American English uses only the British verb form or the spelling for both senses.

UK noun	UK verb	US noun	US verb
defence	defend	defense	defend
licence	license	license	license
offence	offend	offense	offend
practice	practise	practise	practise
pretence	pretend	pretense	pretend

Miscellaneous

Personal pronouns

I is always the subject of the verb and **me** is always the object.

✔ I cooked lunch.

✔ He cooked lunch for me.

This doesn't change if there is more than one subject or object in a sentence.

✔ Pete and I cooked lunch.

✔ He cooked lunch for Pete and me.

If you are unsure whether to use **I** or **me** in this kind of sentence, try it without the other person's name and it will be clear which to use.

✔ He gave top marks to Serena, Keith and me. → He gave top marks to me.

✘ He gave top marks to Serena, Keith and I. → He gave top marks to I.

Myself, yourself etc

All pronouns ending in **-self** or **-selves** are reflexive pronouns and are used only to refer back to the subject of the sentence. They can never be subjects of a sentence themselves.

✔ I treated myself to a new set of speakers.

✔ Mordecai takes himself very seriously.

✔ Where do you see yourself in ten years' time?

✔ Theresa and I googled ourselves and didn't like what we found.

✘ Cassandra and myself crossed the road.

Never use any of these pronouns as a more polite or formal way of addressing people, or if the subject of the sentence is different from the reflexive object (eg **I** must always be paired with **myself**).

✘ If you have any questions, please contact Professor Plantaganet or myself.

Plural or singular?

Use singular verbs for describing faculties, teams, groups etc.

✔ The faculty has voted to keep the building open.

✔ The Oxford Quidditch team has won the first British Quidditch Cup.

To disambiguate when talking about the individuals who make up a group, use 'the members of' or 'each member of'.

✔ The members of the faculty are mostly absent during vacations.

✔ Each member of the faculty has an email address.

Addresses, phone numbers, websites etc

URLs

Omit **http://** unless the URL does not begin with **www** and omit any trailing slash at the end of the URL, unless the URL does not work without it – check before you omit (but ensure that any links in online documents retain the **http://** so that they point to the correct place).

For secure websites, include the **https://**

✔ www.ox.ac.uk/gazette

✔ https://www1.admin.ox.ac.uk/councilsec/governance/committees

✘ www.ox.ac.uk/gazette/

Referring to webpages

Only capitalise the first word (and any proper nouns), but consider instead using the actual URL for disambiguation in print or hyperlinking the descriptive text. **Never** change the capitalisation within a URL as it may cease to work.

✔ For the cost of placing an advert, see the *Gazette* website's *Classified advertising* page.

✔ For the cost of placing an advert, see www.ox.ac.uk/gazette/classifiedadvertising.

✘ For the cost of placing an advert, see www.ox.ac.uk/gazette/Classifiedadvertising.

email addresses

These are case sensitive in the part before the @: *Gazette@admin.ox.ac.uk* is not the same as *gazette@admin.ox.ac.uk*. In practice, the majority of ISPs ignore this distinction, but consider carefully whether to use upper case if required to avoid ambiguity (eg between lower-case *l* and number *1*).

phone numbers

Use spacing between parts of numbers (international code, area code, phone number) to make it easier to read.

Include full area code (eg 01865) for national publications.

✔ To contact us, ring Bill on 01865 778899.

Include international code for international publications (eg +44 1865).

✔ To contact us, ring Bill on +44 1865 778899.

Mobile numbers follow the same format as above.

✔ To contact us, ring Bill on 07777 778899.

✔ To contact us, ring Bill on +44 7777 778899.

For local/internal publications, give extension numbers and include in brackets the number to dial from outside the University phone network.

✔ To contact us, ring Bill on (2)78899.

street addresses

Give full address, including postcode, when writing to a primarily non-Oxford audience (postcodes allow people to find buildings more easily online).

✔ The event takes place at the Church of St John the Evangelist, 109A Iffley Road, Oxford, OX4 1EH.

If writing for a local audience (eg advertising a concert), the name of a well-known building is enough on its own.

✔ The event takes place at the Sheldonian Theatre at 8pm on 12 October.

Acknowledgement and thanks

Our grateful thanks go to:

- Dr Serenhedd James, St Stephen's House, for advice on ecclesiastical matters, general proofreading and the splendid example at the end of p19.
- Linda Loder for giving such engaging presentations of such dry subject matter.
- Joanna Bason and Andrea Roger for writing such a helpful guide to addressing titled individuals and allowing us to use it and refer others to it.
- Nadja Guggi for designing and typesetting the guide so patiently.
- All the members of the style guide review panel, past and present.

If you have any queries about using this guide,
please contact:

Public Affairs Directorate
University of Oxford
Wellington Square
Oxford OX1 2JD
gazette@admin.ox.ac.uk

B Planning & Design Example

CS261: Planning & Design Report

Technical Description

Purpose of the Program

Our software will allow a user to track the software development project they are managing and also creates a risk assessment (initial and as the project progresses) which is evaluated using several metrics (hard and soft). Risky areas and suggestions on how to mitigate these risks for any software project will also be highlighted. This will be done through a website which can be accessed by the project leader. This design document illustrates the preliminary planning and design of our software. We will provide a description of processes to give an idea of how our software will work along with justification for our design choices. An outline of the user interface, subsystems interaction, testing plans and also risk assessments will be covered. Naturally, some specifics of the final product may differ from what is described but the general architecture and design should be similar.

Language and Libraries

Our choice of language was primarily motivated by our team's familiarity with Python and web technologies (Javascript, HTML/CSS), the compatibility between these languages, and appropriate libraries. Additionally, Python is a typical choice for ML applications, and we will be using SQLite for its interoperability with web libraries. The libraries are as follows:

- **Flask** - web development library made for Python 3.
- **WTForms** - created with compatibility with Flask in mind, this library makes creating web forms significantly easier if they include specific validators.
- **SQLAlchemy** - database object relational mapper, that is, databases can be expressed as objects in Python 3.
- **Werkzeug** - security library used for things such as password hashing.
- **PyGithub** - allows our project to retrieve GitHub repository data. Used as a metric.
- **sklearn** - Machine learning classification and regression toolkit library.
- **Pyfpdf** - PDF generation library. Will be primarily used to create the project risk and data graph output.
- **Matplotlib** - Library that is used to generate graphs to present data to the manager.

Description of Processes

Database Schema

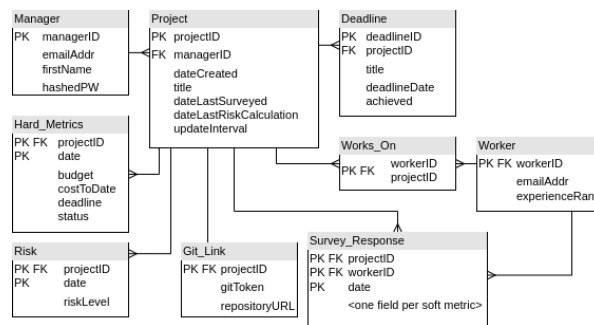


Figure 1: Database Schema

Manager Table The manager table is accountable for keeping manager accounts registered within our system. All passwords are hashed for security.

Project Table

This table is created if a manager creates a new project. Each project is associated with its creator, the manager, thus the foreign key is the managerID. Every project has an updateInterval that stores the frequency of our emailed surveys and the frequency of GitHub pushes within that interval. dateLastSurveyed and dateLastRiskCalculation are stored to enable periodic pushes of the survey and updates to risk calculation respectively.

Hard_Metric Table

The hard metric table stores all initial hard metrics after the creation of a new project. Creating a separate table for hard metrics allows our system to update and create new rows easier when implementing the form within our website. Furthermore, the separation makes it easy to implement new hard metrics in case of an update in the future.

Risk Table

The Risk table stores the calculated risk from our algorithm and associates it with a project, along with its date of calculation to track dateLastRiskCalculation in the Project table.

Git_Link Table

This table was chosen to be separate as gitToken, the GitHub token, can expire and be changed. Separating the table makes pushing the form easier. ProjectID was chosen to be the primary and foreign key to create a one-to-one relationship with Project table.

Deadline Table

This table stores the internal deadlines of a project that the manager creates. Each deadline is represented as a row.

Worker Table

This table stores all the software engineers that are associated with our system. Note that experience can vary from the same software engineer depending on their role. As such, the primary key is the workerID instead of the emailAddr, allowing them to change the experienceRank depending on the project.

Works_On Table

The Works_On table links the Worker Table with the Project Table.

Use-Case, Sequence Diagrams and Interaction

We have designed a Use-Case diagram to present all possible actions of key actors in our system, represented by figure 2 below. Note that the actions available depend not only on the role of the user (Manager/Developer) but also on whether they are signed-in or not.

The interaction of our system is as follows:

Log-in/Sign-up

A manager logs in or signs up using the login/signup form of our website. A signup would create a new row in the table Managers, whereas a login would check for existing credentials in the table. There is no interaction with the algorithm in this functionality.

Project creation

See figure 3.

Editing project data

Editing project data, such as its existing hard metrics, is similarly explained in figure 3. In particular, once hard metrics are POSTed through the website, a new row in the Hard_Metric table is inserted with data. This data is then fetched by the algorithm to re-calculate the risk. After the re-calculation, the new risk is stored in the Risk table and can be displayed in the website at will.

Survey

The survey interaction of our system is split into an if statement. If the software engineer fills the survey for the first time, the website prompts an initial survey. Once the initial survey is filled into the website and stored in the database, the website prompts back to the software engineer with the soft metric survey. Once the soft metric survey data is POSTed, this data is stored in the database under the Survey_Response table. This data is then fed into the algorithm, which re-calculates the risk, stores it in the database, which then can be viewed by the manager. If it is not the first time, the steps explained above repeat, except that they do not fill in the initial survey.

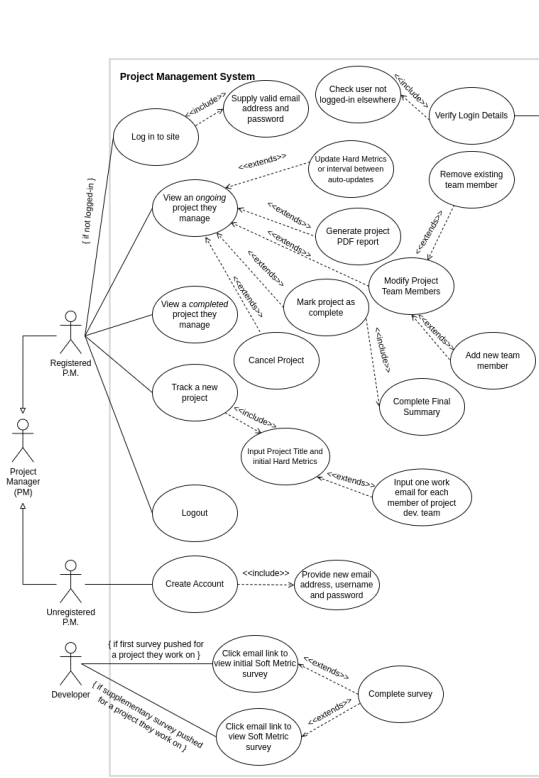


Figure 2: Use-Case diagram of our system

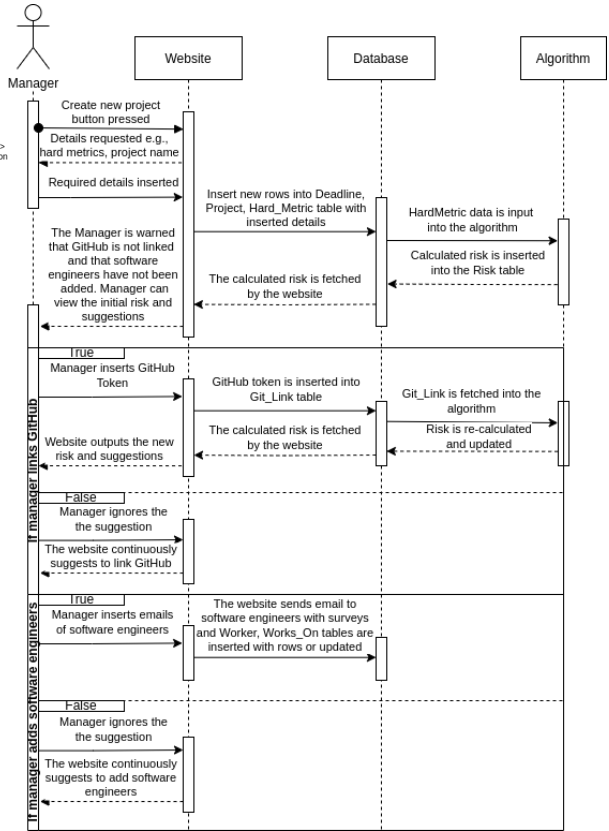


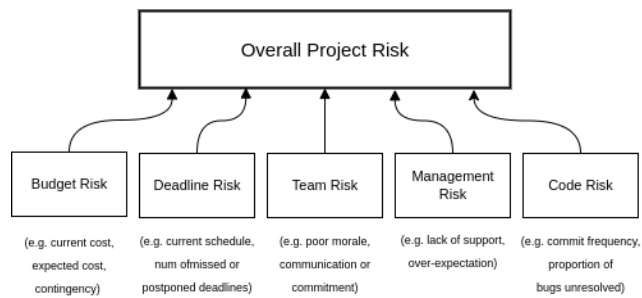
Figure 3: Sequence diagram of creating a project

Risk Assessment Model

The role of the Risk-Assessment Model is to receive a snapshot of the project metrics at a specific point in the development (the independent variables) and to return a value indicating whether the project is likely to be a success or a failure (the dependent variable). Given the complexity of such a relationship, we decided it would be most effective to use a form of Supervised Learning, where the model is trained on labelled data (I.e. a set of projects at various points in their development which have been labelled as ultimately being a success or failure). Furthermore, each project which is tracked to completion/failure using the software provides a new row of training data which can be fed-back into the learning model, to improve the accuracy over-time.

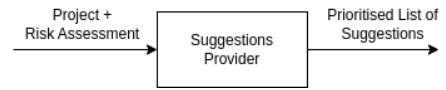
For this problem, there are two main types of Machine Learning model which are applicable: Classification and Regression. Initially, we planned to implement a Classifier such as Support Vector Machines or Decision-Trees in order to obtain a binary (Success/Failure) response, with the confidence of the model representing a measure of the likelihood of the predicted outcome. However, our research found that Logistic Regression models are more appropriate for this problem having previously been applied successfully for similar prediction tasks (Salem, Rekab, & Whittaker, 2004). Specifically, while traditional Linear Regression models produce a continuous output value, Logistic Regression models produce a discrete classification (e.g. 0 or 1, corresponding to success or failure) (Kanade, 2022) which is naturally more suited to the Risk-Assessment Model.

Additionally, to meet requirement **R.4.1**, our system must provide suggestions which can be taken to reduce the risk level, so it is important that the Risk Assessment can be decomposed into components. As such, the model will be divided into five sub-models, each of which provides a risk estimation for a different element of the project (Budget, Deadline, Team, Management and Code). This design offers greater granularity to the Risk-Assessment, so suggestions can be provided according to which component(s) are predicted to fail.



Suggestions

We have decided that suggestions will be provided by a dedicated Suggestion-Provider object, which takes a Project and its Risk Assessment as input and returns a list of suggestions. The actual implementation will be primitive as it will be hard-coded to return suggestions for each of the five categories which fails in the Risk Assessment. Our main reasoning is that because many projects suffer from similar issues, suggestions can be generic enough to apply to multiple projects.



User Interface

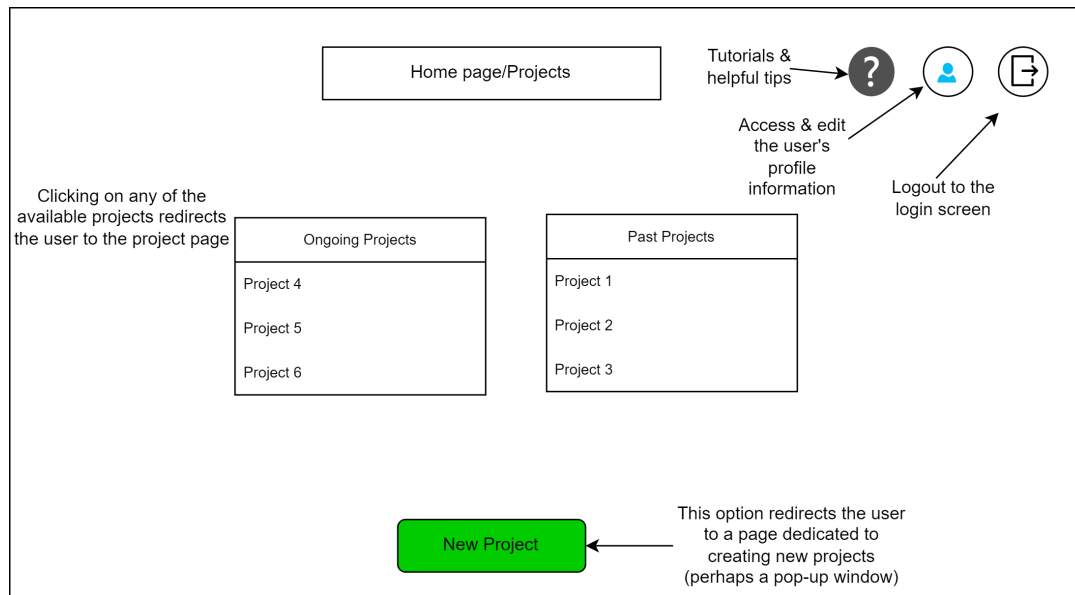


Figure 4: Project Management Interface

This is the design of the home page in which the user may select to view the details of current or past projects

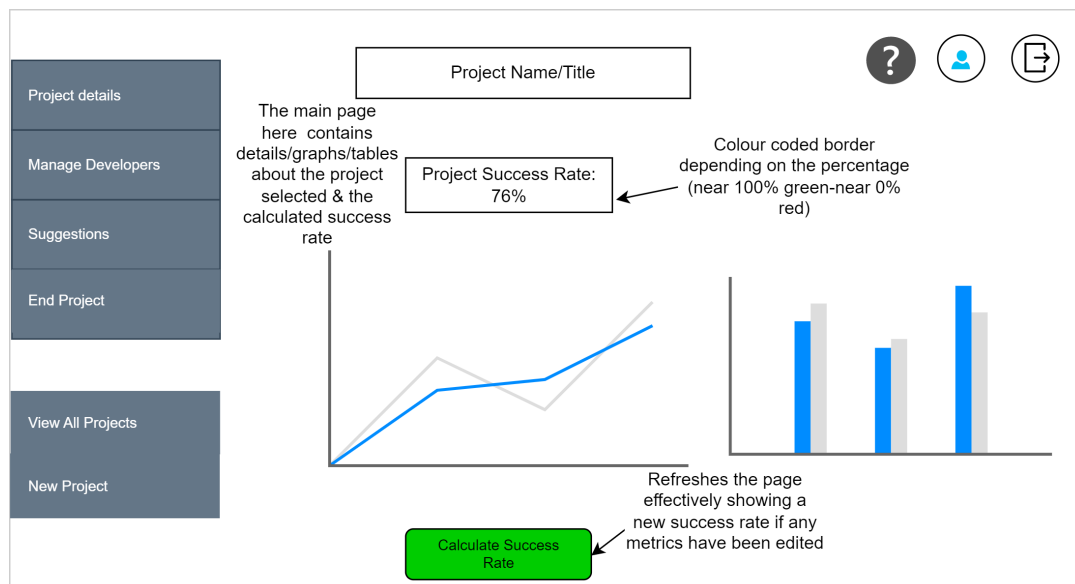


Figure 5: Project Risk and Statistics Page

Upon clicking on any project, the user is redirected to that project's main page.

The menu provides options through which the user may change the project metrics which should affect the output provided. The output is in the form of tables, graphs, success rate percentages and general suggestions. If any of the metrics have been changed (either from "Manage Developers" or "Project Details" sections), the website automatically refreshes showing updated results. The "Completed" projects will have these modification options greyed out and unclickable, so they cannot be interacted with.

UI Design Choices

The website and the user interface have a concise and minimalist design. It will consist of approximately nine different pages some of which will be represented as pop-up pages or lists. The UI is designed to be clear and have a high affordance to the user using tables, lists and menus wherever possible. Two icons (the account management and the log-out icon) can be found consistently on the top right of every page that supports said features. Additionally, at the bottom of each page there is a button of the same colour that indicates the "interact" button, providing a primary action for that page (e.g. to calculate the success rate, to create a new project or to continue to the next page when choosing which project to interact with). The affordance is provided by the simple icons located in any areas that the user may interact with. In case the user requires assistance, a third icon on the top right of any page that may need it, contains an "information/FAQ" clickable option. Lastly, the main use of the website which is to give suggestions and calculate a success rate of a software project is located in a clear and convenient place within the website so as to minimise user interaction when performing simple tasks.

GitHub

To provide an effective analysis of the hard code metrics (commit frequency and bug tracking), it was deemed appropriate to link the user's GitHub repository to our software to access this data. The python library PyGithub contains the necessary functionality to collect information from the user's specified GitHub repository. A user will need to provide a Personal Access Token, with at least full repo permissions and the URL of the repository in order to successfully sync and extract the required data. Tokens can also be made to expire, so whilst the user should provide a token that will last for the project's duration, deadlines may be altered, and a user will need to provide a token once again.

GitHub has a feature known as GitHub Issues where team members can report any bugs or improvements and mark them as completed when necessary. It also tracks user commits, so the frequency of commits can be extracted. Milestones are another feature of GitHub that let users set deadlines for the project; however since these are not mandated and may not be used by a project, it was deemed appropriate to instead collect this information from the user of our software for reliability.

Survey

Survey Questions

We have determined that measuring soft metrics in our software is necessary to provide an accurate estimation of the project failure risk. Since soft metrics in essence are not quantifiable data, we will collect this data via surveys which contain approximately 5 questions with a 1 to 5 scale (where 5 indicates a higher chance of success for all questions, thus standardising the data for the algorithm) to make the metric measurable. The survey will be emailed to all team members to ensure we receive input from all relevant stakeholders.

Initial Survey

The initial survey is unique, gathering data from before the software developers have started working on the project. We also require team members to let us know their expertise when they join a project; this is a hard metric, but we take advantage of a survey being sent out to gather this data.

Some examples of survey questions that may be asked (bold indicates the metric being assessed):

1. I agree with the initial project requirements & design planned out by the team - **Degree of project planning**
2. I fully understand the goals and objectives of the project that the customer is asking for - **Degree of project planning**
3. Choose the rank which best describes your level of experience as a software developer (**Hard metric - expertise**)
 - Rank 1 - junior software engineer, 1 – 3 years, 2+ languages
 - Rank 5 - senior software engineer, 5+ years of experience, proficient in most modern languages as well as have other team management skills.

Periodic Survey

On the other hand, standard surveys will be sent at regular intervals or whenever the manager decides, to the development team to obtain metrics related to the current state of the team.

Some examples of the survey questions would be (bold indicates the metric being assessed):

1. I feel that the top management is committed to the success of this project and feel comfortable approaching them with any issues regarding the project - **Top-level management support**
2. I feel committed to the success of this project - **Team commitment**
3. I feel that the team communicates effectively - **Team communication**
4. I am happy and confident working on this project - **Team Morale**

Frequency of Risk Assessment

The risk assessment will be calculated after the initial metrics are provided by the manager and the developers. Throughout the duration of the project, the assessment will be recalculated after every change in the data for the project: such as a budget change, or the periodic surveys being completed. If a change occurs, this recalculation will only happen once at 00 : 00 GMT. The user is also able to click a button on the website that recalculates the risk and provides new graphical representation and new suggestions (if a change in the risk is present). Alternatively, if no change occurs or no recalculation happens over a certain period of time, the system automatically updates as well.

The reason such limitations are in-place is to ensure the user does not flood the system with high numbers of requests, which would cause multiple database rows to be created for the same date as well as being demanding on our system.

Visualisation

When viewing an ongoing project, users will be presented with the overall risk of failure as well as several graphs presenting the trends for budget, risk and deadlines as mentioned in the requirements. The graphs will be created using the Python library matplotlib.

We decided that plotting every value would be infeasible for long projects which last over hundreds of days, so the system will instead divide the entire project time-frame (from start to deadline) into a fixed number of intervals. For each interval, the system will then plot the highest and lowest value for the risk over that period. This approach will ensure the overall trend is visible, while limiting the resolution of the graph to avoid overwhelming the user.

At the beginning of the project, the graph will be empty as there is no data to plot, but the time frames will already be decided. If a deadline is moved (which is highlighted in our requirements), then the time frames, and the respective data points will then be recalculated and replotted. This means that a graph that was previously fully-plotted may now be only half full depending on how much the deadline was extended by.

These data visualisation techniques increase the usability of our system and improve the chances that a user will be able to quickly and successfully extract important information about a project. A user is more likely to interact successfully with our website if it presents the risk in a graphical way rather than a text-based way and therefore more likely to come back and use our system again.

Testing

The test plan is laid out according to components of the software (database, user interface etc.) and will be carried out during each sprint cycle ensuring that all components completed during that cycle work as intended.

Initially, we will use preliminary static testing to ensure that the code meets the requirements and to catch any errors. These tests will be performed by other members of the team. Then, we will use Unit-Testing to validate the functionality of specific components and functions, as well as Integration-Testing for interoperability of components (for example, the website login and the database). Given that the majority of the system's back-end will be written in Python, we will be using the pytest library for unit-testing. Pytest is a well-documented (Krekel, 2023) test-suite for constructing and running automated tests and is preferable to unittest for the ease with which new test cases can be defined without needing to define a dedicated Test class.

Testing Plan

Database

Our system’s functionality relies on user and project data being stored and retrieved correctly from the persistent storage (i.e. the SQLite database). We can presume, based on the wide usage of SQLite and its documented testing process (SQLite Contributors, 2022) that the database integrity will be maintained both between queries and when database operations are interrupted (for example, if power is lost). Therefore, we must focus our tests on ensuring our own SQL Functions, Procedures and Triggers are valid and robust.

For example, consider the function to retrieve the mean value of the soft metrics for a given project over a specific interval: *getMeanProjectResponses(projectID, startDate, endDate)*. We will Unit-Test this function by simulating the contents of the Survey_Response table:

Table 1: Example Survey_Response Test State

Project ID	Worker ID	Date	Metric 1	Metric 2	Metric 3	Metric 4	Metric 5
11	1	2020/01/01	4	3	1	1	1
11	2	2020/01/03	3	3	5	1	4
12	1	2020/01/03	1	4	1	5	2
11	1	2020/01/08	2	3	2	4	1

Then, we can assert that the expected result for a call with (*projectID* = 11, *startDate* = 2020/01/01, *endDate* = 2020/01/07) will be a tuple of means: (3.5, 3.0, 1.0, 3.0, 2.5), following the same order as the columns in the table above. Notice that to achieve this result, the function must calculate the mean of only the first two rows; essentially, this test verifies four correctness properties:

1. The mean of each column is calculated correctly
2. Row 1 is *included* (since its date falls on the boundary of the time period)
3. Row 3 is *ignored* (since it applies to Project 12, not 11)
4. Row 4 is *ignored* (since its date falls outside the time period)

Risk Assessment Model

The Machine-Learning model is difficult to test precisely, since it cannot be expected to produce the correct classification every single time. However, we can require that its performance meets a given accuracy threshold; in particular, we can require that the F-Score (Pedregosa et al., 2011), a measure of the overall precision and recall, is at least 80%. Similarly, sklearn provides methods to generate the Confusion Matrix of the model, which indicates the proportion of mis-classified observations for each pair of classes.

Suggestion System

In order to test the Suggestions Provider (SP), we will first need to have fully tested the Risk-Assessment Model (RAM) and validated that its output is appropriate. Then, to test that the Suggestions are selected correctly, we will write a series of Unit-Tests simulating different Risk-Assessment configurations (for example, one project where all categories are successful except for Budget; another project where all are successful except team). For each scenario, the development team will consider the list of suggestions and identify those which are deemed appropriate for the test-case.

Additionally, due to the relationship between the RAM and SP, we will perform Integration-Testing to ensure that both components interact correctly. This will be achieved by designing a series of sample projects and manually identifying appropriate feedback, before feeding each of them into the RAM-SP pipeline one-by-one, and verifying that the returned suggestions are appropriate for the given sample.

Website and UI

Testing the general functionality of the website is a very high level procedure. A team member other than the person who wrote the front end code will attempt to input erroneous data as well as navigate the website in “unorthodox” ways. Some such ways could be to log out of an account and then press the “back” button on the top left of the browser, attempting to log back in that way. Another detail of the website mentioned in requirement **R.1.2**, i.e., logging a user out from an older device when the same user attempts logging in from a new device, can be trivially tested.

Usability testing will also be carried out by non-technical users. These acceptance tests will highlight the ways in which different users may perceive web-page elements. Non-technical users will be asked to complete a series of tasks and later asked to review how they felt using the website. This ensures that the website is built in such a way that is usable and easy to understand for all users despite having different technical capabilities, fulfilling requirement **R.5.1**.

Static UI tests will be performed to ensure that all features work as it is intended. For example, the usage of the “question mark” icon will be tested to ensure that it is able to guide non-technical users to be able to use the website with minimal training and works as intended, ensuring that requirement **R.5.3** is met.

As mentioned in requirement **R.5.4**, we will mainly test the website on Google Chrome version 109.0.5414.119 on Windows 10. However, the website should be compatible with any other browser that supports Bootstrap 5 and its respective Javascript version. Testing this will be a matter of running the website on different browsers (for example, Firefox, Safari, Microsoft Edge) and using the features of the interface that use Bootstrap or interact with JS. Any observations will be well-documented.

We will also be testing how the website renders on different screen sizes such as mobile phones and laptops to check requirement **R.5.5**. Load testing will be carried out where different scenarios are stimulated to check the behaviour of the system under different throughput by the user to ensure requirement **R.5.6** is fulfilled.

GitHub Linkage

To ensure our software receives the correct information regarding the project’s code, it is important to check if our software successfully obtains this data (**R.2.2**). By setting up a sample repository, we can perform unit tests between our software and the repository such as:

- After opening nine issues, each with one of the 9 different labels, run the `open_issues_count()` function to check all issues are being detected
- Using `get_issues(state = 'open')`, iterate through all results and display them, comparing them against the issues open in the repository and check they are identical
- Set the repository to Private, and check the provided token still has access to the repository

Process Documentation and Risk Assessment

We have chosen an Agile methodology, inspired by Scrum. Due to the nature of our development, we do not have full access to the client and so we could not fully implement Scrum. We have chosen to enact weekly cycles, each following these steps: sprint planning, the sprint, and the sprint review. An initial product backlog, consisting of the components necessary to the software has been constructed. This will evolve as the project progresses as more information is gathered about how the team works best and any further functionality the software may need. During sprint planning, this backlog will have the most pressing items taken from the stack (for example, establishing the database is crucial as many future components will interact with it) and broken down into sub-tasks that can be feasibly completed in a cycle, such as the production of a schema, or the database triggers. Each task also requires testing so these will also be allocated in the cycle to be designed and implemented. Once the plan is agreed upon, the sprint can begin, in which each team member works to complete their allotted tasks. To ensure progress is made, daily scrum meetings will be held to track everyone’s progress and ensure the tasks are completed by the end of the cycle. Once a cycle is finished, all completed tasks can be removed from the product backlog. Finally, the review is when all the team members can demonstrate their work to the rest of the team, and any comments and improvements can be added to the backlog to be addressed in the following cycle. In order to promote the smooth completion of cycles, we have elected a Scrum-Master who will be responsible for leading the sprint meetings (planning, daily and review).

The following is an outline of what each cycle hopes to address, but as each cycle progresses, these events will naturally be broken down further and completed at different rates as some subsystems are more complex than others:

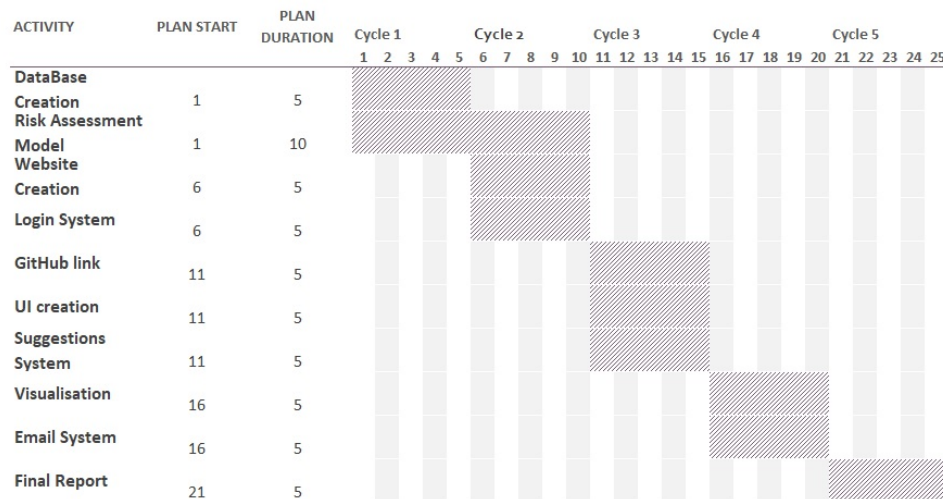


Figure 6: Cycle Gantt Chart

Risks

Risks have been ordered by decreasing potential impact to the development. The inherent risk refers to the threat-level posed when no effort is made to mitigate the danger whereas residual risk is the remaining threat after a treatment has been put into place. We also provide an explanation of how each risk applies to our team (Wilson, 2022).

Risk	Summary	Inherent Risk	Treatment	Residual Risk	Explanation
Size Underestimate or timescale not estimated correctly	Underestimating the complexity of the project and requirements which creates a risk of going over the deadline	High	Ensure a clear timeline is developed over time, with each task broken down into manageable sub sections that are understood by everyone	Medium	Due to our minimal experience with projects of this size, it is hard for us to reliably estimate the complexity and time-scale correctly. However, we have focused on designing a system which can be implemented with the knowledge and tools available to us
Not completing cycle(s) on time	If the end of the cycle(s) is reached without finishing all the tasks outlined in the cycle timeline	High	Have a member of the team assigned to track the progress of each cycle and ensure all the developers are completing the tasks	Medium	As students, we are not developing the program full-time and therefore can fall behind on the plan we have set out
Tool Underperformance	Libraries or software we intend to use are not performing the way it was expected.	Medium	Perform appropriate research before introducing new tools. This should happen in the design stage of the project.	Low	

Specification delays/ change in requirements	The client not presenting the specification in time or requirements are constantly changing.	Low	The agile methodology we have chosen allows us to assess new requirements at the beginning of each cycle and decide if implementing them is feasible.	Low	The client has already presented the specification to us. There is a chance we will change our requirements as we start development however this can be planned for at the beginning of each cycle.
Change in group members	Team members leaving the group and a new person joining who is unaware of the aims of the project.	Low	Re-allocate the roles such that at least two people are responsible for each component.	Low	If any team member leaves, for example in our case if someone drops the module or leaves the university; we will have redundancy in the form of at least one other member able to work on each feature and explain the requirements to a new member.

Overall, we believe that the risk of our project is manageable; we should be able to complete all the tasks, and meet the requirements if we plan for the risks and work towards mitigating them as outlined in the table. During each weekly meeting, we will continue to reassess the risks and ensure that any issues are managed correctly.

References

- Kanade, V. (2022, Jun). *Linear regression vs. logistic regression: Understanding 13 key differences*. Retrieved from <https://www.spiceworks.com/tech/artificial-intelligence/articles/linear-regression-vs-logistic-regression/>
- Krekel, H. (2023, Jan). *pytest documentation*. Retrieved from <https://buildmedia.readthedocs.org/media/pdf/pytest/latest/pytest.pdf>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Salem, A. M., Rekab, K., & Whittaker, J. A. (2004). Prediction of software failures through logistic regression. *Inf. Softw. Technol.*, 46, 1-5.
- SQLite Contributors. (2022, May). *How sqlite is tested*. Retrieved from <https://www.sqlite.org/testing.html>
- Wilson, F. (2022, Sep). *Inherent vs residual risk: Differences and examples explained - ntask*. Retrieved from <https://www.ntaskmanager.com/blog/inherent-vs-residual-risk/>

CS261: Requirements Report

Team Organisation

Roles

For a team of 6 people, we have assigned the following roles:

- **Project Manager** - Andreea Nicolae
- **Research** - Raihanah Lukman
- **Software Engineer** - Heath Nicholson, Aris Papakonstantinou, Billy Pentney, Cem Yilmaz

This decision was made based on an initial assessment of each member's strong points and how they feel they could contribute best to the project. The project manager is responsible for organising the team and making sure deadlines are met. Additionally, the researcher's role is to bring valuable information to the group ensuring that any decisions made in design and requirements are backed-up with research. The development team was naturally divided based on everyone's individual skill and past experience.

Methodology

Our methodology is an agile approach inspired by Scrum. We decided we wanted to work in cycles, with each cycle achieving a set of requirements, similar to the scrum cycles. A member of the development team will take a similar role to that of the Scrum master and ensure the deadlines are being met. However, since we are all undergraduate students with little experience, we could not fully implement Scrum since we are not familiar with it. Additionally, due to the nature of the project, we are unable to interface directly with the client, so the scrum methodology would not work. Instead we have opted for creating a plan for the development with weekly cycles that we can check the progress of. This works for us since we have a clear deadline and the requirements are already set out from the beginning.

Important decisions with regards to the project are generally taken democratically to ensure the engagement and agreement of all team members. Meetings were agreed to be organised with a minimum of 2 meetings a week, every Monday (online) and Friday (in-person). We have also agreed upon having small or urgent meeting opportunities when required.

Functional Requirements

Account Management

R.1 - A manager must be able to create and login to/logout from a password-protected account with the system using their email address.

R.1.1 - Each user should have access to the following utilities:

- Create multiple projects
- Mark a project as completed, or terminate it
- Add or remove deadlines for sections of the project, and mark them as met or not
- Add or remove the emails of team members for the given project, for data-collection purposes
- View the project metrics stored by our software for the project and receive any suggestions the software has
- View all the projects they manage

R.1.2 - If a user is logged in on a device and attempts to login on another device, the attempt should be denied and a prompt will be displayed to log out of the first device.

Metrics

R.2 - The software will measure a number of metrics about each tracked project in order to generate the risk assessment and provide appropriate suggested actions.

We have split the definition of metrics into two categories. Hard metrics are objective factors of a project that are prompted as an input during the creation of a project in our software and updated throughout. Soft metrics are defined as factors that are subjective - i.e. project attributes which are dependent on opinions and emotions of members of the team.

Hard Metrics

R.2.1 - The Hard Metrics will be provided by the Manager via user input

R.2.2 - Hard Metrics related to the code can be gathered directly by linking a Github repository

Metric	Definition	Reasoning
Budget	The fund allocated for the project	To generate a budget trajectory To check if the project is on track to be completed with the funds allocated
Bug Tracking	The number of known bugs and issues which are retrieved from the linked GitHub repository	To determine the proportion of outstanding bugs/issues
Commit Frequency	The mean time between commits to the linked Github repository	To determine how frequently the code-base is updated/the activity of the development team
Timeframe	The project's various internal deadlines as well as the final deadline	To track the progress of a project through the number of deadlines met/postponed
Team Size	The number of members in the project's development team	To measure the scale of the project; to ensure the budget is appropriate for the project size
Expertise	The software-development experience of each team member	To measure the sufficiency of skilled software engineers for a project; to ensure the budget is appropriate for the team's skill level

We have decided that the software should not consider code quality, as this metric is particularly difficult to measure and outsourcing the quality-check to third-party software presents a potential security risk.

Soft Metrics

R.2.3 - The data for soft metrics will be collected via short periodic surveys which must be completed by each member of the team.

- Surveys will be sent out periodically with the option for the manager to send out additional surveys.

R.2.4 - When a new team member is added to the project by the manager, the team member will be sent a unique, initial survey, to acquire information about their experience and expertise.

Table 1: Soft Metrics Table (Ahimbisibwe et al., 2015), (Ewusi-Mensah, 2003)

Metric	Definition	Reasoning
Top-level Management Support	How the team members feel regarding the top management's commitment to the success of project and how involved executives are with team members	To measure the relationship of the company with the project, including investment and significant of the outcome. When upper-management is more involved, the project may be subject to more scrutiny and higher expectations, leading to pressure and chance of failure
Team Communication	The level and quality of communication between team members to avoid issues or tensions in the group	To determine how well the team discusses and cooperates to ensure a successful project. A healthy and productive discourse between team members is more likely to result in constructive collaboration
Team Commitment	The dedication that a team member has to the success of the project	To measure how dedicated the team is to the success of the project. High Commitment means that an employee is likely to put in additional effort, going above and beyond to ensure the project meets its expectations
Team Morale	The overall attitude and satisfaction of the team members in the workplace	To measure the satisfaction of the team as overworked and unhappy employees are more likely to make mistakes or be reluctant to the commit to the project

Degree of Project Planning	The extent to which the project's requirements are thought-out and understood by the team	To measure how well the specification has been translated to a complete set of requirements and how well the team members are aware of the aims of the project
----------------------------	---	--

Risk Assessment

R.3.1 - When a project is first created by the manager, the system's Risk-Assessment model will provide an estimation of project riskiness based on the initial Hard Metrics.

R.3.2 - After each Manager update to the project, the system will identify new data for the Hard Metrics and update the Risk-Assessment accordingly.

R.3.3 - When the Risk Assessment is generated, the manager should be presented with a list of suggested actions which could be taken to reduce the risk, and this list is ordered by importance and relevance to the project.

R.3.4 - When the Risk Assessment is generated, the manager should be presented with a list of suggested actions which could be taken to reduce the risk, and this list is ordered by importance and relevance to the project.

R.3.5 - When a project is marked as complete by the manager, our system will prompt the manager to input a final summary, evaluating the success of the project.

- The response provided by the Manager will be fed back into the Risk-Assessment model, to influence the assessments it provides for future projects managed via the software.

Visualisation of Output

R.4.1 - When viewing a project, the system will display tracked project metrics along with providing appropriate suggestions.

- The suggestions will be classified as High Priority and Low Priority based on the risk assessment.

R.4.2 - Project Analytics are presented visually, with graphs for:

- Timeline / tasks deadline (stacked bar chart, similar to a gantt chart)
- Budget trajectory (Line graph with forecast)
- Risk-level, over-time (Line graph)

R.4.3 - The project statistics and suggestions can be downloaded as a PDF file which can be shared with the rest of the team or the client if needed.

Non-Functional Requirements

R.5.1 - The software must be accessible via a website with a succinct but well-labelled user interface. That way, the software will be intuitive and accessible for non-technical users.

R.5.2 - A user must only be able to create an account with a password that has a minimum of 8 characters, contains at least one capital letter and number.

- If the password is not strong enough, then the user will be unable to click on the button and an alert will appear.

R.5.3 - The software will require little user interaction other than clicking and entering text making it as easy to use as possible. Furthermore, an "information" or "question mark" icon may be conveniently placed within the page consisting of helpful tutorials or tips for new users. This way, users can receive the minimal training they might need when using the software.

R.5.4 - The system must be accessible via Google Chrome version 109.0.5414.119 running on Windows 10 10.0.19045 Build 19045

R.5.5 - Consequently, it should run in other modern browsers and could also provide support for mobile-friendly resolutions

R.5.6 - The load time of the website should be under one second depending on the host's throughput. The expected uptime of the website should be above 95% meaning that the website will almost always be up unless unprecedented circumstances occur.

References

- Ahimbisibwe, A., Cavana, R., & Daellenbach, U. (2015, 02). A contingency fit model of critical success factors for software development projects: A comparison of agile and traditional plan-based methodologies. *Journal of Enterprise Information Management*, 28, 7-33. doi: 10.1108/JEIM-08-2013-0060
- Ewusi-Mensah, K. (2003). *Software development failures: anatomy of abandoned projects*. MIT Press.