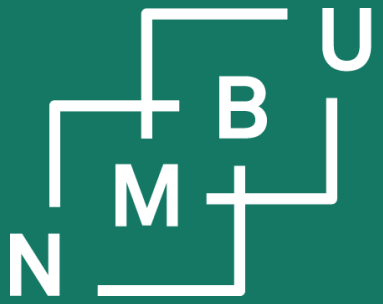
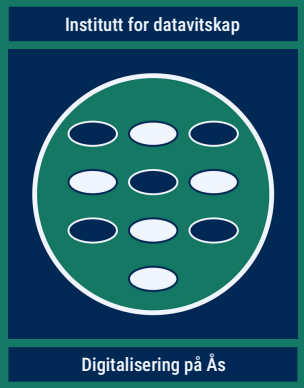


Noregs miljø- og biovitskaplege universitet



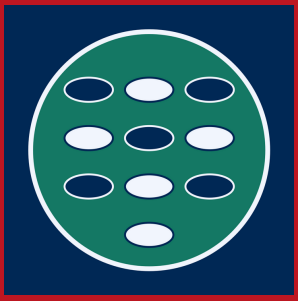
INF203

June advanced programming project

1 Introduction

Noregs miljø- og biovitskaplege universitet

Institutt for datavitskap



Digitalisering på Ås

1 Introduction

1.5 Requirements analysis

1.6 Good documentation

INF203

4th June 2025

Agile requirements analysis

The book by Cohn (2004) is a classic source for agile requirements analysis.

Main points of the approach include:

- Requirements analysis based on an ongoing dialogue with people.
- Explicit diversity in the sort of people that might have requirements.
  - It is not always possible to actually interview and talk to representatives of all groups – in this case, the methodology forces the developers to at least think what requirements they might have.
- General good practice in formulating objectives: They must be SMART. Specially, there must be a clear criterion to see whether they are met.
- Suitable for agile style of development; focus less on static specifications and more on what something is needed for – maybe a new, better way can be found.
- "Black cat or white cat, it does not matter. But it has to catch the mice."

Key concepts for the approach

User story: "Describes functionality that will be valuable" to someone (Cohn). It has a format like "as a <persona>, I want to do <action> to achieve <aim>."

- Good user stories are *independent*, *negotiable*, *valuable*, *estimable*, *small*, and *testable* (INVEST).

Epic: A coarse-grained objective. One way to distinguish epics and user stories is to look at the time that will be required to create the functionality.

- ISO 6515:2018 defines an epic as a "major collection of related feature sets broken down into individual features or user stories and implemented in parts over a longer period of time".

Persona: "An imaginary representation of a user role" (Cohn). It is good practice to talk to actual people and take down their stories, but still present them in an aggregated and anonymized way.

(For this programming project, it is not expected that you interview people.)

Requirements register

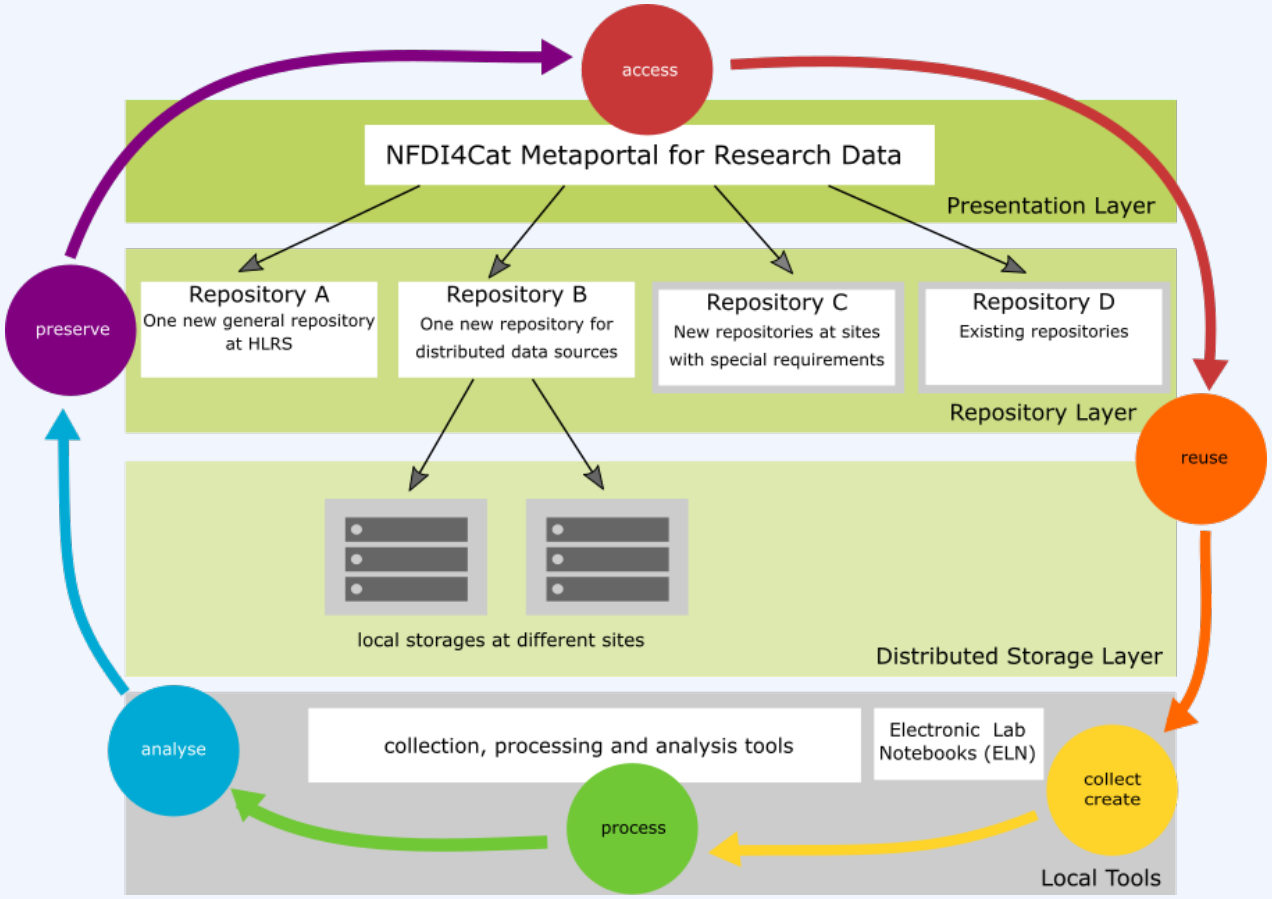
Start building a requirements register now. Maintain it throughout your project. This helps you identify concrete development tasks and, in the end, evaluate to what extent you have accomplished your goals.

A simple way is to write down the requirements in spreadsheet form.

Columns could be, for example:

- Label of the epic or user story
- Design target (what element of your architecture is it for)
- Persona (what abstract representation of a stakeholder has the need)
- Actionable objective (what is it that X wants to be able to do)
- Overarching aim
  - For a user story, this is the epic under which it is subsumed
  - For an epic, this is an even higher-level eludication
- How many working hours would it take to implement and test?
- Priorization: Must, should, could, or won't (MoSCoW)
- Who took the task? Has it been implemented? Has it been tested?

Example: Use in NFDI4Cat (2020/21)

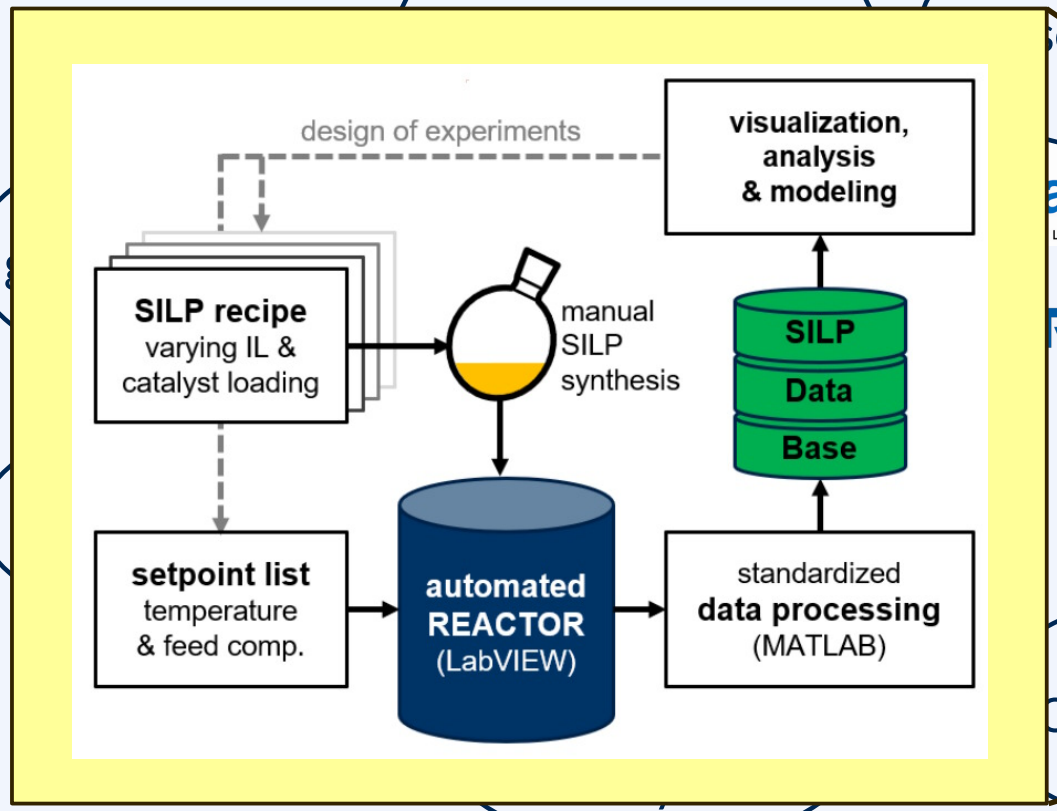


Platforms to be designed:

- Overarching central infrastructure, including the NFDI4Cat metaportal
- Bespoke local repositories
- Generic solution for local repositories

NFDI4Cat

Example: Use in NFDI4Cat (2020/21)



interview 11:

interview 10:
cher-Tropsch process

atalysis
Leibniz-Institut für Katalyse e.V.

interview 17:
FURTHRmind

RWTH AACHEN UNIVERSITY

FAU

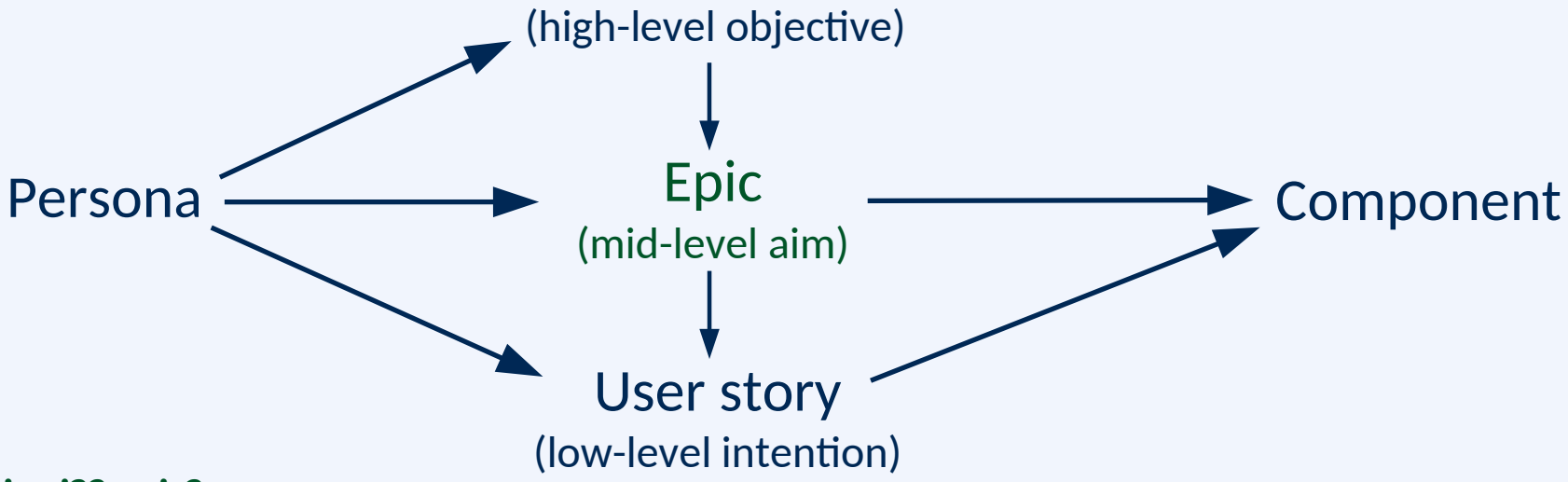
interview 16:
Water-gas shift reaction

interview 15:
Oligomerization reaction

Epics: Design-level objectives

Agile reqs analysis based on **user stories** and **epics**. Example from NFDI4Cat:

Hierarchical structure of requirements in agile software engineering

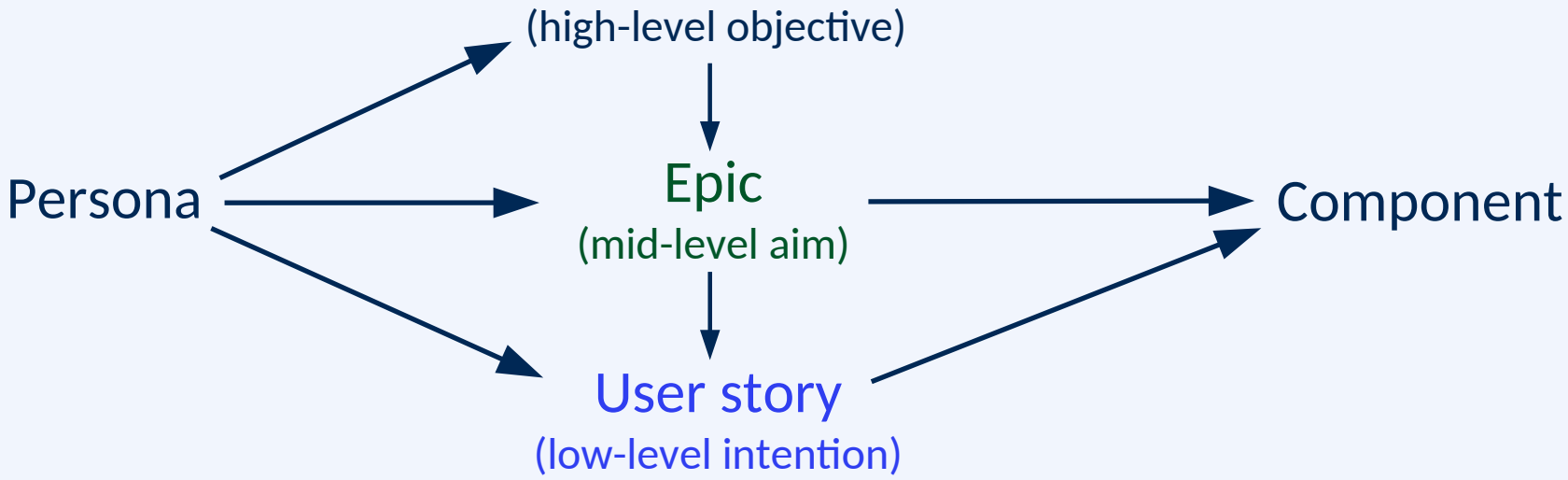


Epic ui??:epic?
As Domain Scientist,
I aim at employing the local XXX knowledge base, which is presently being constructed on the basis of XXX, as a local repository within a federated NFDI4Cat architecture,
subordinate to my objective to create synergies between NFDI4Cat and other ongoing work.

User stories: Implementation-level objectives

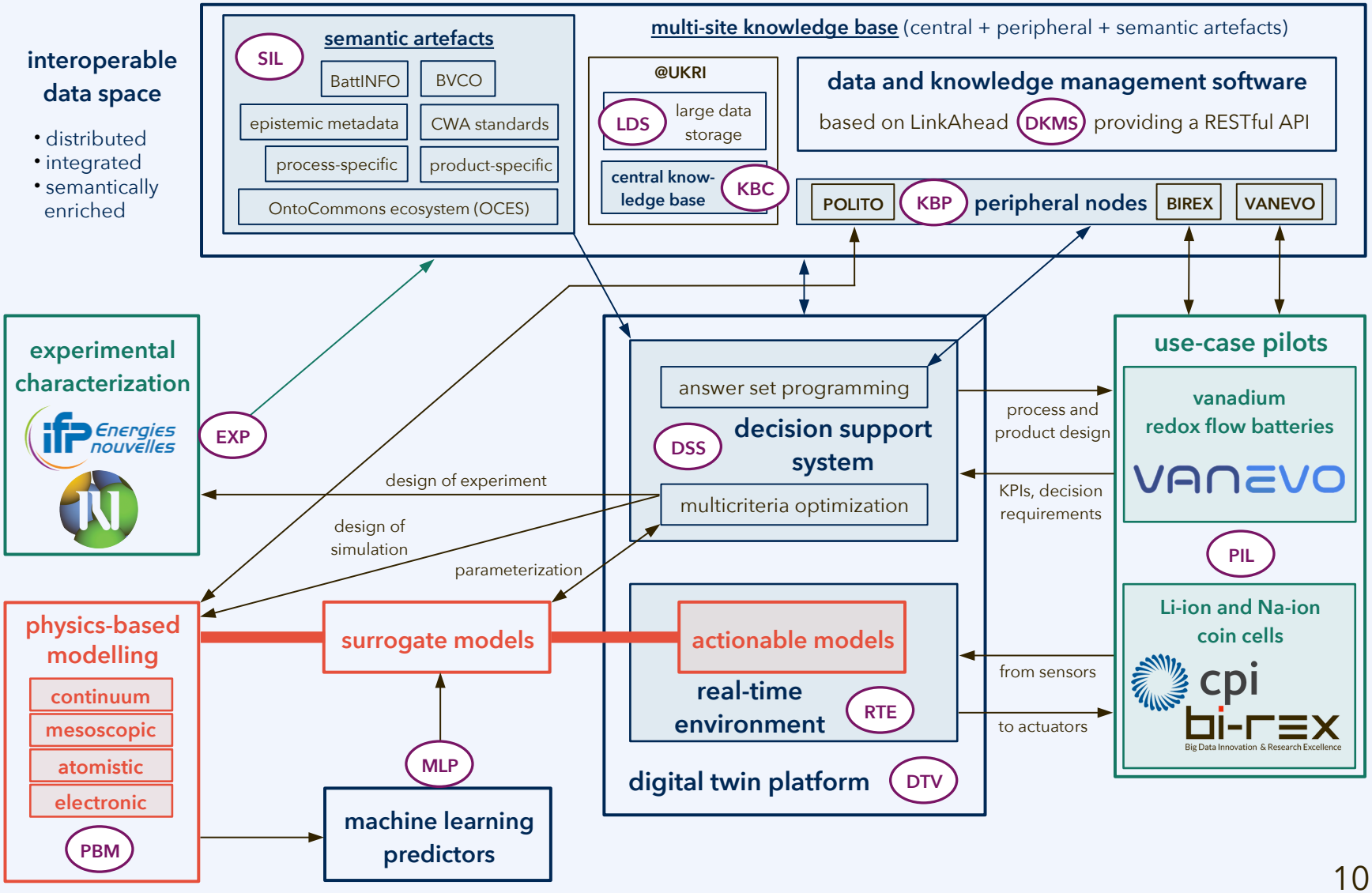
Agile reqs analysis based on **user stories** and **epics**. Example from NFDI4Cat:

Hierarchical structure of requirements in agile software engineering



User story ui??:epic?-us?
As Domain Scientist
I intend to have an agreement on minimum requirements for query languages, if possible including property-graph based approaches, to be implemented by NFDI4Cat Components.
subordinate to my aim of building and operating a local repository within NFDI4Cat.

Example: Use in BatCAT (2024)



Example: Use in BatCAT (2024)

The following tasks conducted internal & external **stakeholder interviews** as part of an **agile requirements analysis** jointly, with task T4.1 taking the lead:

T4.1: "Knowledge infrastructure requirements analysis" (lead: NMBU, contrib.: AAU, DTU, IS)
T4.3: "Data and metadata landscape" (lead: UKRI, contrib.: CPI, IFPEN, NIC, SIMULA)

- T4.1&3 deliver **D4.1**, "Data landscape & infrastructure related requirements," by **M9**.

T6.1: "Industrial & use-case requirements analysis" (lead: VANEVO, contrib.: BIREX, CPI, DTU)

- T6.1 delivers **D6.1**, "Use-case requirements for validation," by **M9**.

T7.2: "Citizens' role and societal & gender dimensions" (lead: NMBU, contrib.: DTU, POLITO)

- "positive and potentially adverse aspects of the societal (incl. citizens', gender) dimensions of the impact, conducting an agile requirements analysis from early on"

We proceeded in the following stages:

1) Preparatory **first-stage interviews** (30 minutes), exchange of ideas.
2) **Second-stage interviews** (30 minutes), developing concrete user stories.
3) **Half-day workshop** (24.6.) for revision and extension of deduced requirements.
4) Analysis and catalogue of requirements (**deliverables** D4.1 and D6.1).

Example: Use in BatCAT (2024)

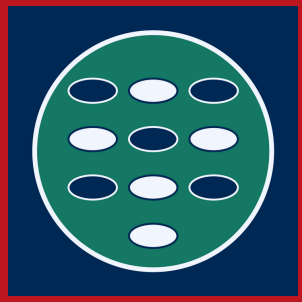
Agile reqs analysis based on **user stories** and **epics**. Example from BatCAT:

We collected & accepted 56 user stories, which are categorized according to:

- **Persona:** What kind of user/developer has such a requirement?
  - Personas: (1) AI: Administrator – internal, (2) DI: Digital twin technology user – internal, …, (8) EE: Experimentalist – external, (9) PE: Policy expert – external
- **Epic:** What is being pursued as an overarching aim?
  - The 56 user stories were grouped into 23 epics.
- **Design target:** What is it for?
  - There were 12 design targets (architecture elements), plus a separate category for non-functional requirements (without specific design target).
- **"MoSCoW" priorization:** "Must", "should", "could", or "will not"?

Example: As a policy expert, in order to enable RFB manufacturers to calculate entries for the digital product passport, I want (myself/manufacturers) to use the decision support system to predict the carbon footprint at process/product design stage. ("S": "Should".)

Noregs miljø- og biovitskaplege universitet



1 Introduction

1.5 Requirements analysis

1.6 Good documentation

INF203

4th June 2025

Comments and docstrings[1, 2]

Follow known good practices when commenting. (We could discuss them.)

Docstrings – a convention in Python – are a special kind of comment, included in the form of a string literal as the first statement within an element.

Recommendation for docstrings:[1, 2]

1) Start with a short summary of the function, method, class, or module. This should be a single sentence that describes its purpose.
2) List and describe each parameter: What it its type, what is its meaning?
3) If there is a precondition that the user is responsible for ensuring (e.g. values within certain range), make it explicit.
4) Describe the return value (if any), state postcondition (if appropriate).
5) If your function/method can raise exceptions, mention it.

Despite all the above, be concise!!
And never let a generative AI tool comment your code.

[1]Python tutorial Section 4.9.7 [2]PEP 257: Docstring conventions

Documentation using LaTeX

Your report/documentation as part of the INF203 submission must be a PDF created using LaTeX.

The exact formatting does not matter very much, you can simply use the document class “article”.

However, there is also a NMBU LaTeX template, which the Data Science and Physics institutes decided to use for master and PhD thesis. With minor edits it also works for other courses.

Template URL: https://github.com/LRydin/NMBU-Thesis-Template

Documentation of generative AI use

Even where generative AI tools are allowed, the following is obligatory:

- "**clearly indicate** how and where AI-based programs have been used".
  - "If the student uses AI-based programs in a submitted assignment or in other work to be submitted for assessment or as a compulsory activity […], the student must **account for the use**, *e.g.*, **in the** […] **methodology chapter**. How the use is described depends on whether AI is used as a writing aid or as a research method."

- "The student is responsible for **ensuring that information** generated using AI-based programs **is reliable**."

- "The student must **critically evaluate** the AI-generated results and ensure that they are based on research or other relevant literature."

Restriction of generative AI use

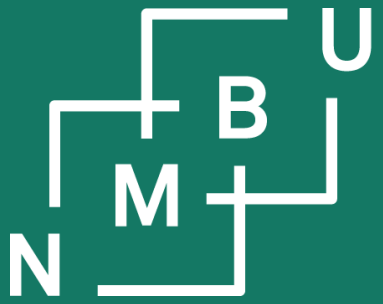
NMBU allows the use of AI-based tools in the preparation of assignments that are part of mandatory coursework or assessments, unless the course description explicitly states that AI-based programs are not permitted.

For didactic reasons, the use of generative AI may be restricted.

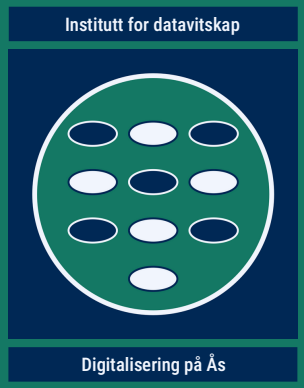
In the case of INF203, **you are not allowed to include any code created from generative AI tools in your submitted programming project code**.

For other purposes, it may be used, under the provisions stated before.

This includes support at creating LaTeX documents, language polishing etc.



Noregs miljø- og biovitskaplege universitet



INF203

June advanced programming project

1 Introduction