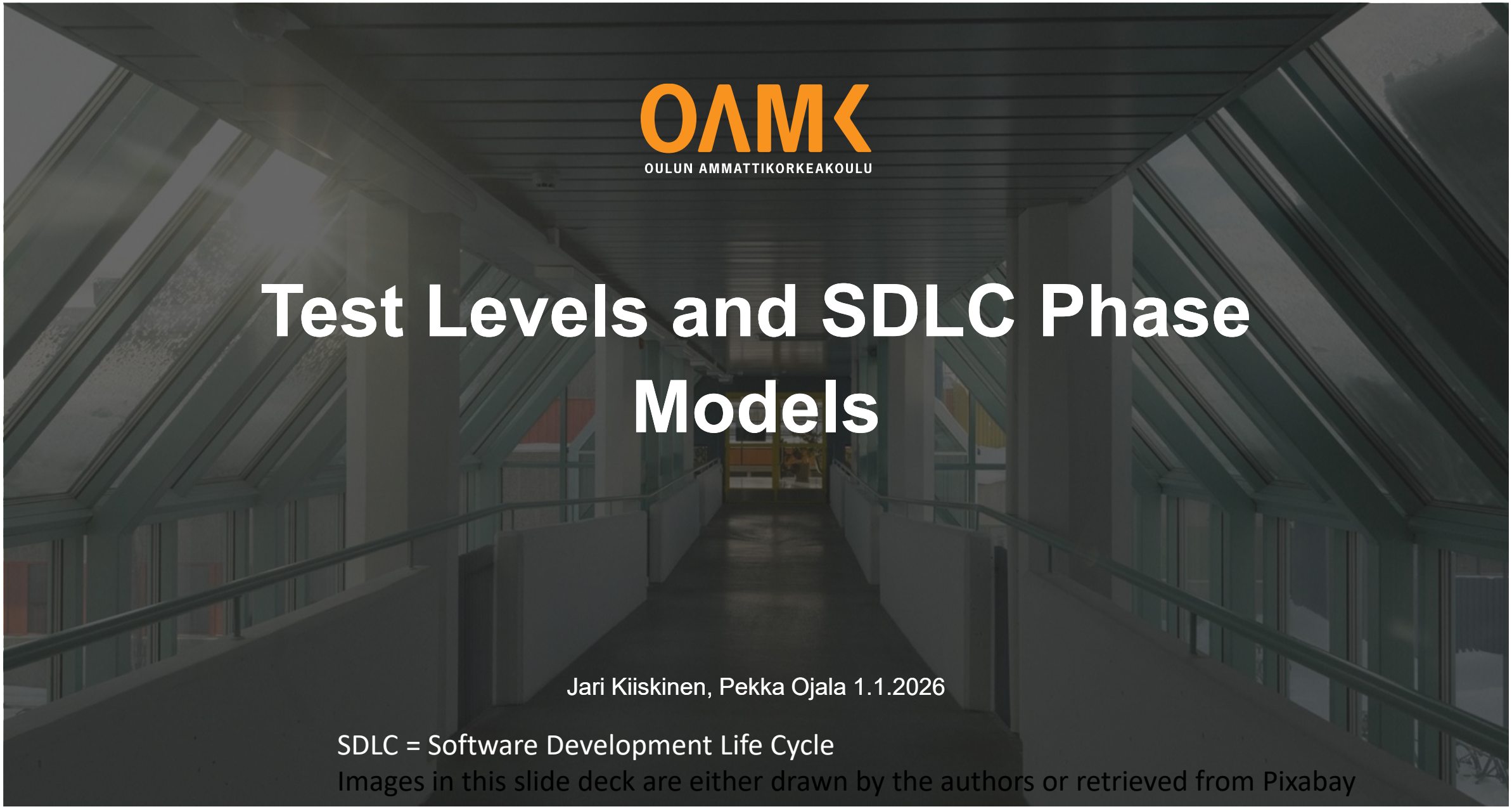


Test Levels and SDLC Phase Models

Jari Kiiskinen, Pekka Ojala 1.1.2026

SDLC = Software Development Life Cycle

Images in this slide deck are either drawn by the authors or retrieved from Pixabay

Contents

Software Development Phase Models

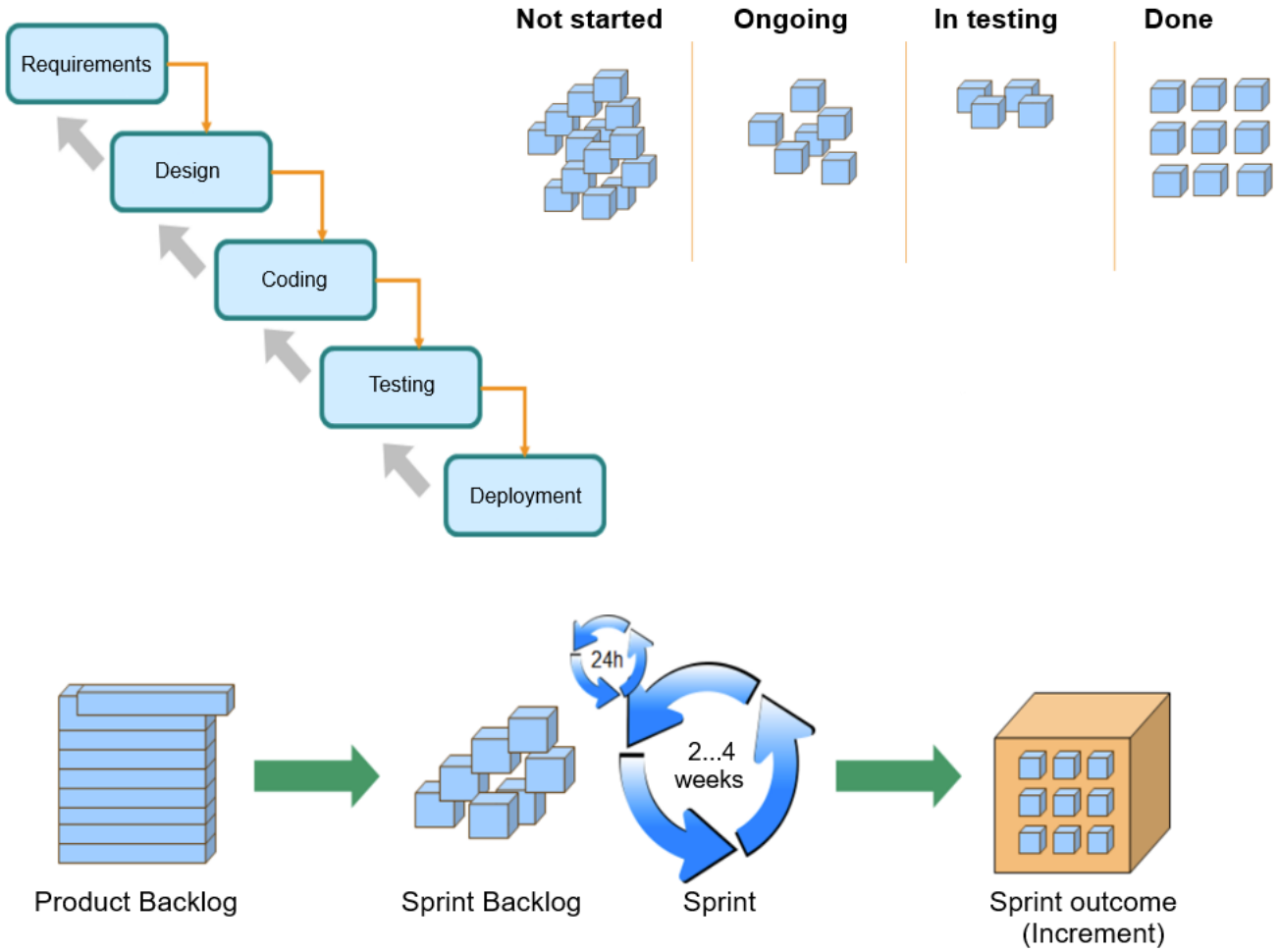
- Waterfall model
- Scrum
- Kanban

Test Levels and Classifications

- Unit testing
- Integration testing
- System testing
- Acceptance testing
- Best practices

OAMK

Software Development Phase Models



Software development uses various phase models or life cycle models (Software Development Life Cycle, SDLC)

Since the phase model significantly affects software testing, the tester must understand how software development occurs in different phase models

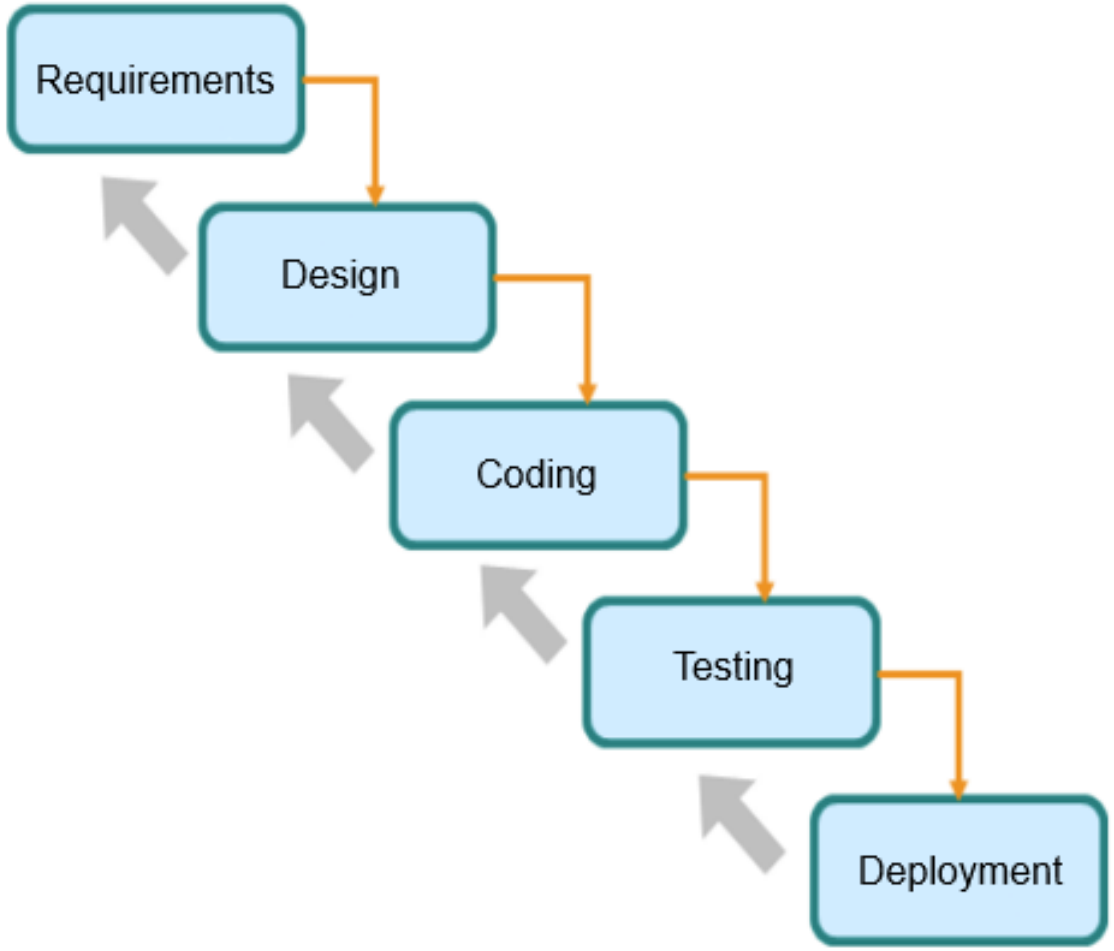
OAMK

Waterfall Model

The waterfall model is the best-known example of a sequential phase model

The main phases of software development are executed sequentially, completing each phase

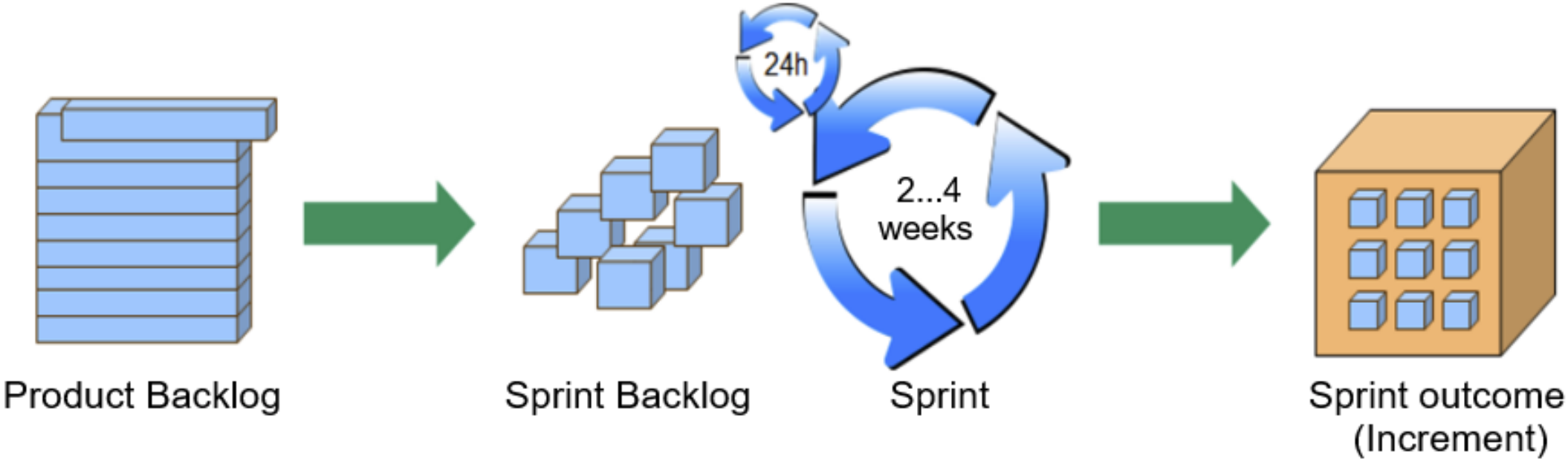
If necessary, return to earlier phases, which can significantly increase development costs



OAMK

Scrum

- An iterative and incremental model of agile software development
- Tasks are taken from the development backlog into time-boxed sprints
- A sprint produces something concrete (demo, software component, etc.)
- Where is the tester, what is the tester's role, where does the testing happen?



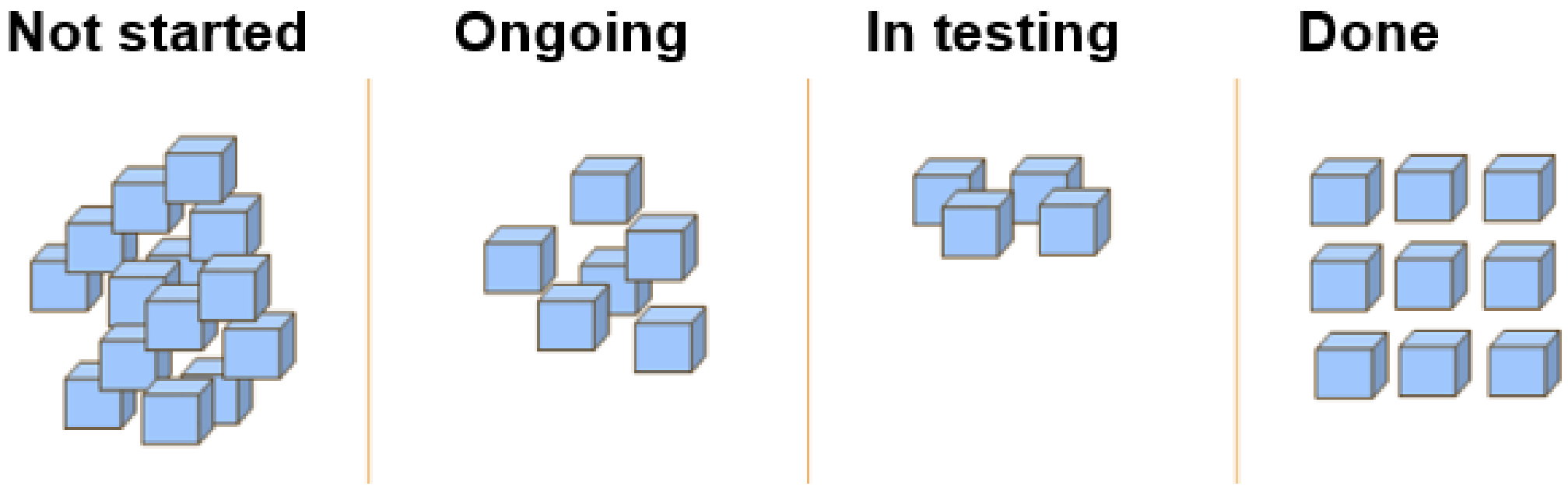
OAMK

Kanban

An iterative and incremental model of agile software development

Tasks move on the Kanban board from left to right

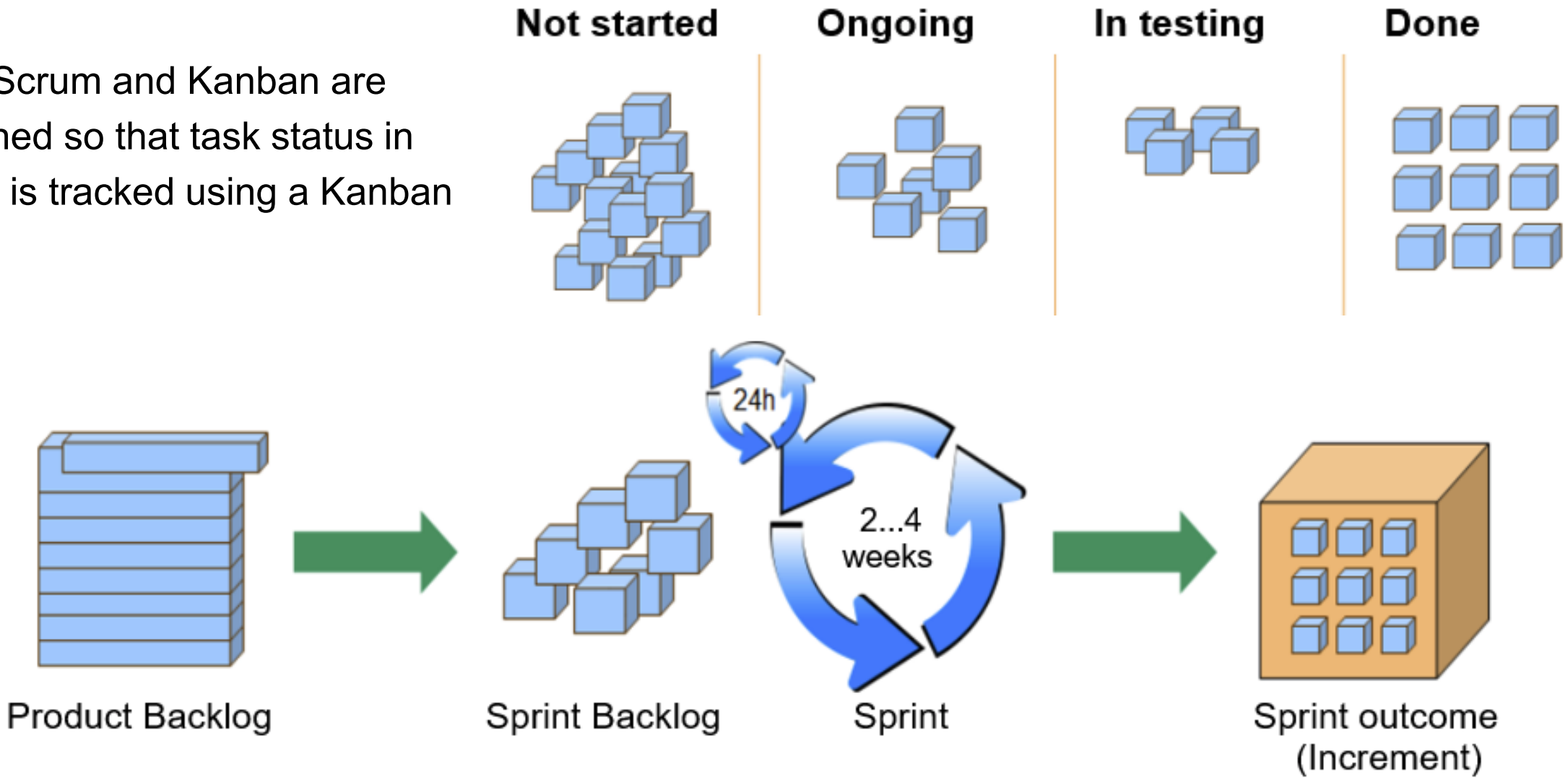
Emphasis, compared to waterfall or scrum, is on ..constant flow?



OAMK

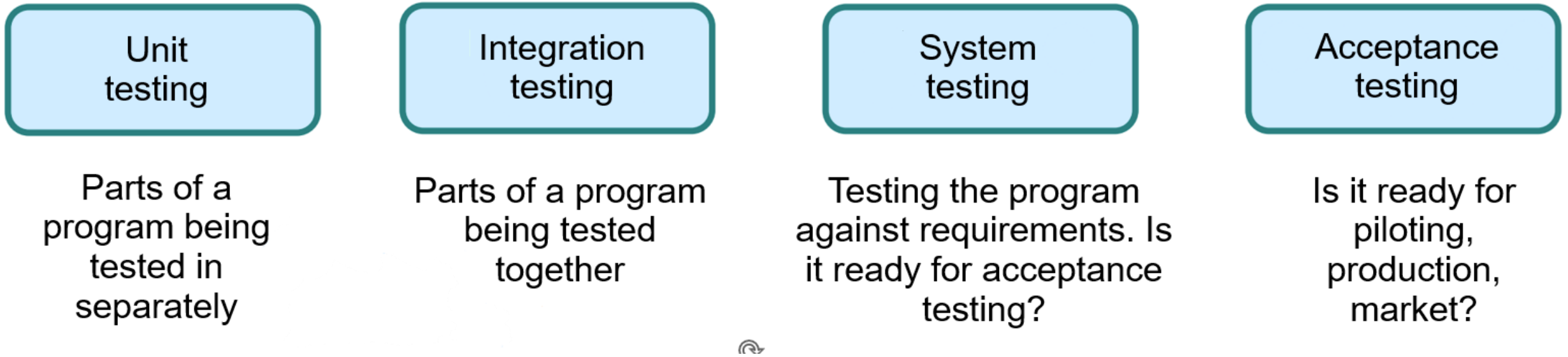
Scrum + Kanban

Often Scrum and Kanban are combined so that task status in sprints is tracked using a Kanban board



Test Levels

- The most common test levels in the industry are



In practice, it's almost always about a **new version** of software (instead of single one-shot release), because during the software lifecycle, several versions will likely be released, and today it's quite common for releases to come at a rapid pace. Short release cycles also indicate that software is being actively developed.

Unit Testing

A Unit is an abstract concept because in software it can mean different things. Typically, a unit performs one or more related functions. Testing focuses on the unit.

The **software classes** visible in the diagram below can be understood as units, but they contain smaller units within them

Petra Programmer

Product

Shopping cart

Peter Programmer

Session

Bank interface

Login

Register

Logout

For example session handling consists of Login, register and logout components

Unit and component Testing best practices, the DOs

- Make unit tests before the implementation, TDD, if possible
- Don’t make too fancy unit tests for simple components
- What to do with dependencies? Databases, UI, other components?
- Bug -> unit test -> fails -> fix -> unit test ok!

Petra Programmer

Product

Shopping cart

Peter Programmer

Session

Bank interface

Login

Register

Logout

For example session handling consists of Login, register and logout components

OAMK

Unit/component Testing pitfalls, the DONTs

- Unit tests are slow and clumsy to execute. Unit tests need to be fast and reliable
- Unit test length is comparable to the actual code size?

Petra Programmer

Product

Shopping cart

Peter Programmer

Session

Bank interface

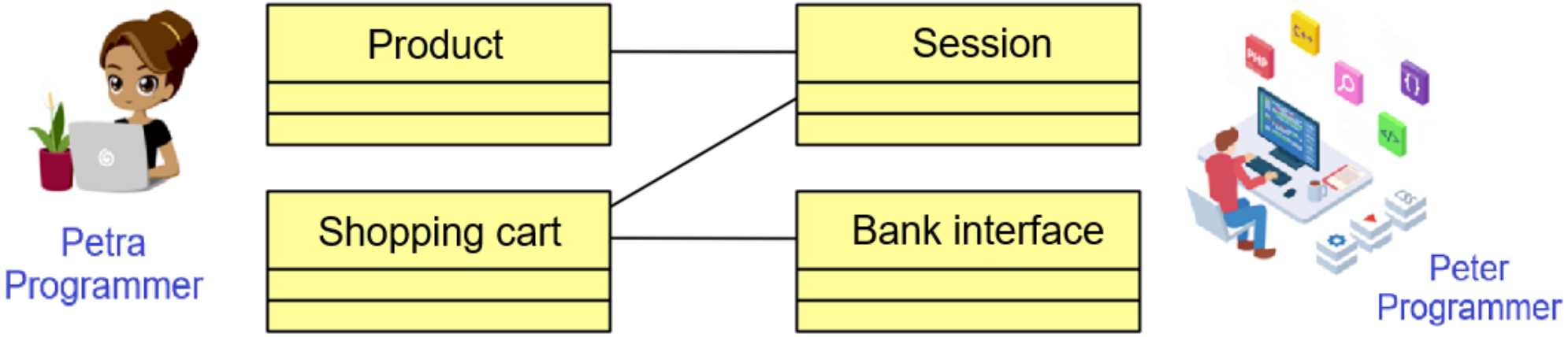
Login
Register
Logout

For example session handling consists of Login, register and logout components

OAMK

Integration Testing

- Integration testing aims to find errors in interfaces and connections between different parts of software, and to combine different parts into functional subsystems and ultimately into a functional whole
- Who's testing: often developers. What are the requirements to execute integration testing?



System Testing

In system testing, the entire software is the test object and it can focus on various aspects such as

Functionality

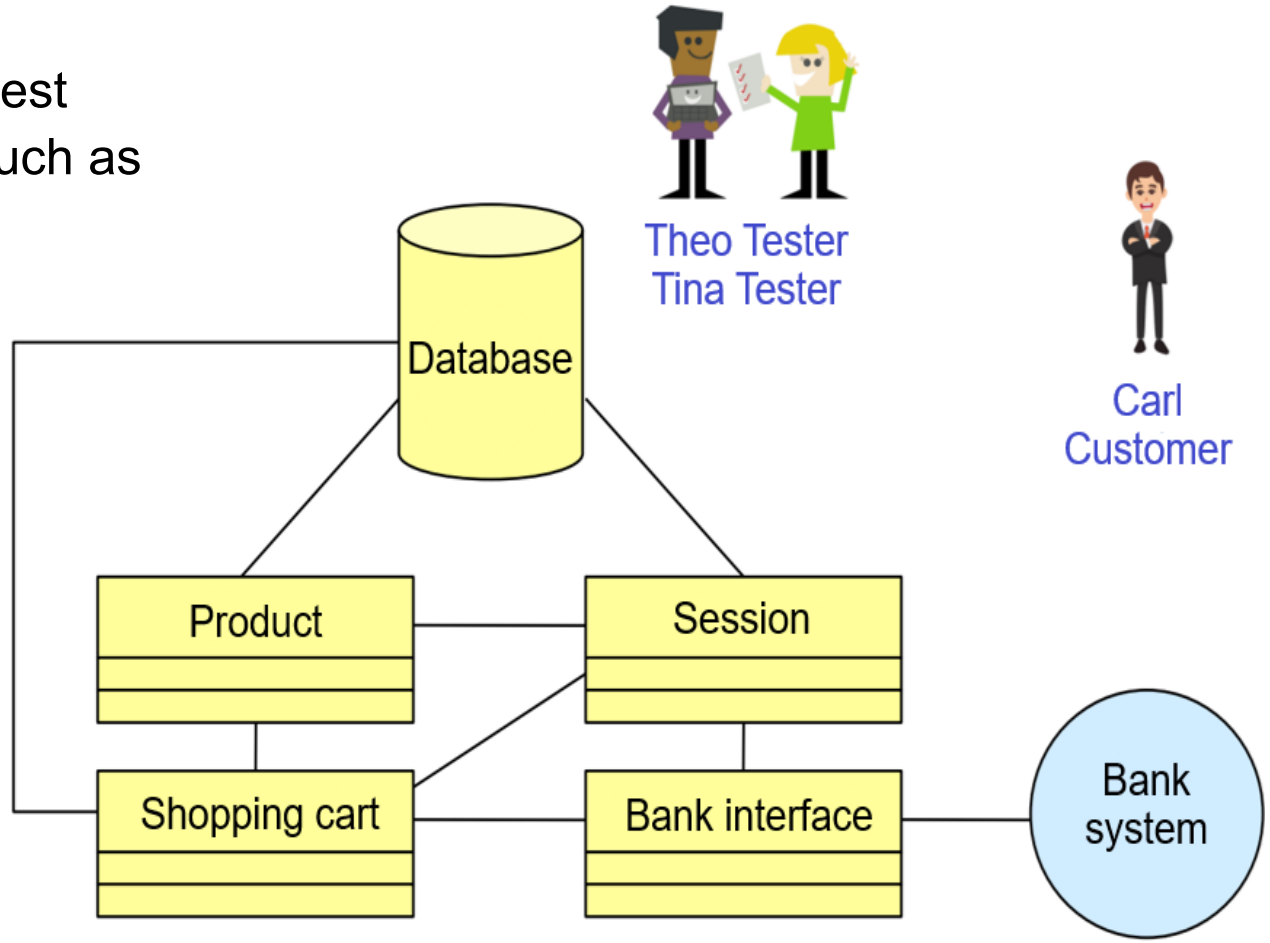
Performance

Security

Accessibility

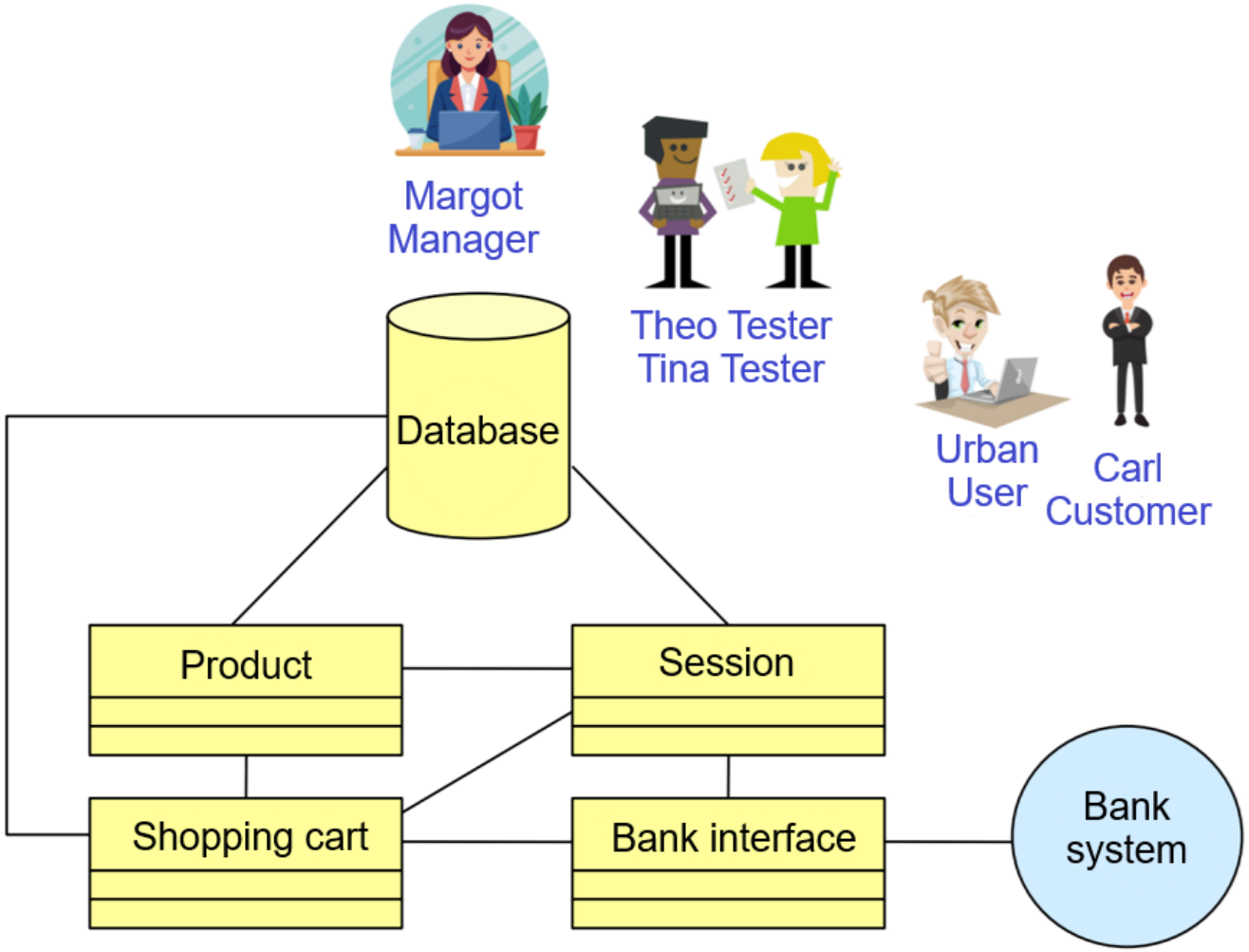
etc...

Testers are typically doing the testing as the software is fully usable now, finally



Acceptance Testing

- The test object is the entire software
- Actual users of the software are doing the testing
  - UAT (User Acceptance Testing)
  - Is the software ready for actual use?



Best Practices 1

The following practices are good regardless of which phase model is used in software development:

- Tests are planned in advance
- Tests are available, i.e., not solely in one person's possession
- If actual testers are not available, cross-testing among programmers is better than programmers testing their own code (blindness to one's own errors and solutions)
- Tests are recorded when necessary so performed tests can be revisited later
- Testing tools and their usage is documented: no documentation -> bad documentation -> good documentation!

Best Practices 2

More best practices regardless of phase model:

- Testing is automated in areas where automation brings clear benefits
- If testing reveals bugs or deficiencies, key personnel (e.g., programmers and higher stakeholders, if necessary) become aware of them
- Since testing functionality alone can require significant resources, careful consideration is needed for where testing is directed
  - Example: If time doesn't allow testing performance and security separately in their own system tests, decide which is more important from users' perspective (both are very important, but sometimes you must choose where to direct more testing resources)

Best Practices 3

More best practices regardless of phase model:

- Future users of the software are involved in testing as early as possible, but not too early (e.g., if software is too incomplete)
- If possible, involve testers who have not had any contact with programmers
- Broad and comprehensive test data

Best Practices 4

- More best practices regardless of phase model:
  - At the latest in **acceptance testing**, testing should be performed in an environment that is either the real environment or as close to it as possible. Expertise helps to decide how close is close enough.
  - If it's not software for a specific client but a mass market product, beta testers should be utilized
  - In Beta Testing, a larger group of testers tests the application in a real environment before releasing a new version

Best Practices 5

- Bootcamps of teams in tricky times to transfer knowledge (and pain).
- Coder-developer pairing physically-online (few hours…days) for synchronized problem solving and testing. Domain knowledge vs code knowledge

Sources and Further Reading

- GURU99. Test Levels in Software Testing. https://www.guru99.com/fi/levels-of-testing.html
- VALA. UAT testing, i.e., acceptance testing: What does it mean and why is it important? https://www.valagroup.com/fi/blogi/hyvaksymistestaus/
- VALA. Integration testing: What does it mean and what are its benefits? https://www.valagroup.com/fi/blogi/integraatiotestaus/
- VALA. System testing: What is it and why is it important? https://www.valagroup.com/fi/blogi/jarjestelmatestaus/
- VALA. Unit testing: What is it and why is it important? https://www.valagroup.com/fi/blogi/yksikkotestaus/
- Koodia pinnan alla episode 10 on testing featuring Finland's top tester Maarit...


OAMK
OULUN AMMATTIKORKEAKOULU