Technical Debt

Group session 9

isabelrl@uio.no



Agenda

- Quiz
- Quick recap
- Exercises
- Previous exam questions

Quiz!

Repetition from lecture

What is technical debt?

- **Short-term** design or implementation decisions that **make future changes more costly**
- Can be intentional or unintentional
- Affects maintainability and evolvability
- Not always negative, sometimes useful for rapid delivery

Principal: The cost of removing the technical debt

Interest: The additional cost caused by keeping the debt

Types of Technical Debt

Type	Example
Code debt	Duplicated code, complex methods
Architectural debt	Tightly coupled components
Test debt	Missing automated tests
Documentation debt	Missing or outdated documentation
Structural debt	Poor modularization or dependencies

Why manage Technical Debt?

- Increases development time and costs
- Makes systems harder to maintain and extend
- Can reduce productivity over time
- Accumulated debt may lead to development crises
- **Prioritization is key: not all debt needs to be removed**

Identifying and Reducing Technical Debt

Identification

- Code smells
- Static code analysis (e.g. SonarQube)
- Architectural analysis
- Code reviews

Reducing debt

- Refactoring
- Reducing unnecessary complexity
- Improving modularity
- Addressing high-impact debt first

Technical Debt Management Approaches

Approach	Goal	Examples
Proactive	Prevent debt	Education, guidelines, culture
Continuous	Monitor and manage debt regularly	Code reviews, retrospectives, static analysis
Reactive	Respond to existing debt	Impact analysis, roadmap evaluation, business cases

Technical Debt Management Maturity

Level	Description
0 – Unaware	No awareness of technical debt
1 – No tracking	Aware, but no structured tracking
2 – Ad-hoc	Informal tracking by individuals
3 – Systematic	Dedicated resources and monitoring
4 – Measured	Automated tools and measurements
5 – Institutionalized	Organization-wide standardized process
6 – Fully automated	Automated, data-driven decisions (research stage)

Exercises

Case 1: the booking platform

A startup developed an online booking platform under a tight deadline. To launch quickly, the team made several shortcuts.

The platform now has:

- Duplicated booking logic across multiple modules
- Few automated tests
- Outdated documentation
- Strong dependencies between booking and payment services

The company is growing, and the team must implement cancellations and refunds next month.

Discuss:

1. Identify the technical debt in the system.
2. What interest might the company pay?
3. Which debt should be prioritized, and why?
4. Was taking technical debt necessarily a bad decision?

Case 1: possible solution

1. Technical debt

- Code debt: Duplicated booking logic
- Test debt: Missing automated tests
- Documentation debt: Outdated documentation
- Architectural debt: Tight coupling between services

2. Interest

- More time required to implement refunds
- Increased risk of introducing regressions
- Additional effort to understand undocumented code
- Changes in one component may require changes in others

3. Prioritization

Architectural debt and missing tests may be particularly important because refunds will interact with existing booking and payment functionality.

The final priority depends on expected interest, risk and the cost of removing each debt.

4. Was taking debt a bad decision?

Not necessarily. For a startup, releasing early can provide valuable feedback and reduce business uncertainty.

However, the debt must be managed before it becomes too expensive.

Case 2: managing debt in an agile team

A software team works in two-week Scrum sprints. Developers regularly complain that the codebase is difficult to maintain.

The team has several problems:

- Technical debt is not documented
- New features are always prioritized over refactoring
- Code reviews are rarely performed
- The Product Owner does not understand the impact of technical debt
- Developers occasionally fix debt on their own initiative

Discuss:

1. What technical debt management maturity level best describes the team?
2. Suggest one proactive, continuous and reactive approach.
3. How could the team include technical debt management in Scrum?
4. How could developers convince the Product Owner to prioritize technical debt?

Case 2: possible solution

1. Maturity level

Level 2 – Ad-hoc tracking

Developers are aware of technical debt, but management depends on individual initiative rather than a systematic process

2. Approaches

Proactive: Provide training and establish development guidelines

Continuous: Introduce regular code reviews and debt tracking

Reactive: Evaluate existing debt against the roadmap and create a business case

3. Scrum integration

- Include technical debt items in the Product Backlog
- Discuss debt during Sprint Planning
- Use retrospectives to identify recurring problems
- Reserve capacity for refactoring where appropriate
- Make technical debt visible to the Product Owner

4. Convincing the Product Owner

Communicate the business impact rather than only describing technical problems.

For example, explain how technical debt increases development time, delays features or creates risks for planned functionality.

Previous exam questions

2019

1.3 IN5140 H19 Exam Task 1.3

What is the interest of technical debt?

- The extra-negative impact paid because of the presence of the technical debt
- The effort required to fix the technical debt
- The investment that companies do to manage technical debt
- The technical issue that needs to be solved

2024

Technical Debt approaches

Select the correct alternative from the drop-down list:

Continuous approach, Proactive approach, Reactive approach

1. Creating a business case for technical debt is a Select alternative .

2. Facilitate seminars and workshops on how to manage technical debt is a Select alternative .

3. Evaluating the roadmap before refactoring technical debt is a Select alternative

1p bonus if all are correct ones are chosen.

2024

Technical Debt approaches

Select the correct alternative from the drop-down list:

1. Creating a business case for technical debt is a Select alternative ✖ (Continuous approach, Proactive approach, Reactive approach) .

2. Facilitate seminars and workshops on how to manage technical debt is a Select alternative ✖ (Continuous approach, Proactive approach, Reactive approach).

3. Evaluating the roadmap before refactoring technical debt is a Select alternative ✖ (Continuous approach, Reactive approach, Proactive approach)

Have a nice weekend! :D



isabelrl@uio.no