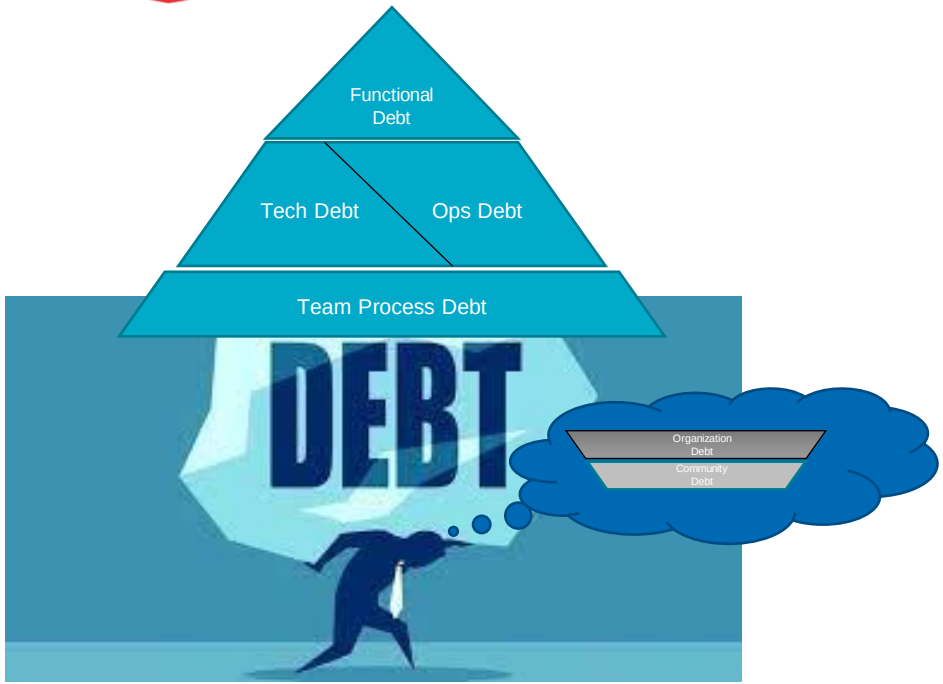
AMIS | CONCLUSION

NL.OUG
NL ORACLE USER GROUP

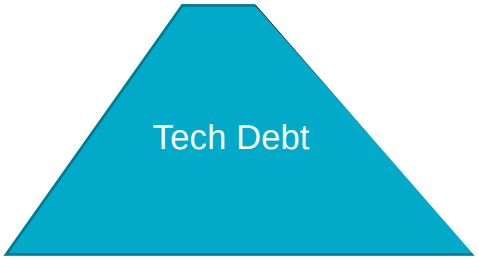


Scaling [down] the Mountain of Debt –

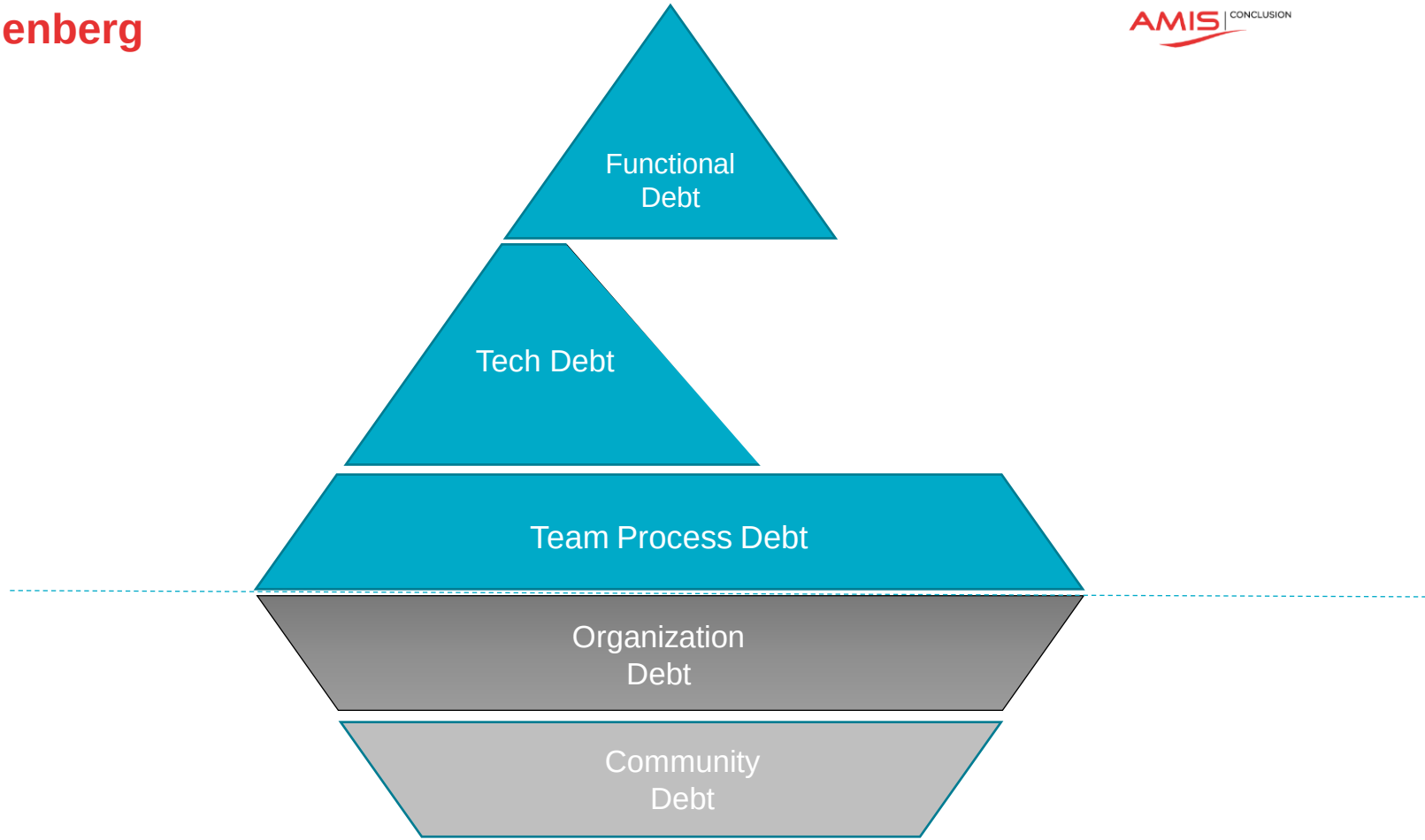
Four dimensions of IT Debt

Lucas Jellema
nlOUG, 13 Januari 2022

Tech Debt

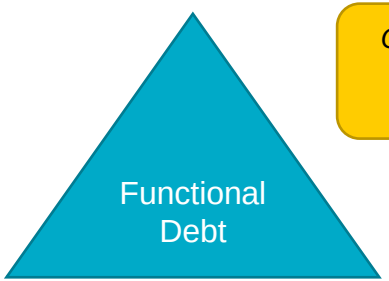


Schuldenberg



Functional Debt

- Some of the desired behavior and features are not available
- Specific situations and conditions are not supported
  - errors, unknown behavior
- Style deviations (color, font, layout)
- Accessibility shortcomings
- Specific browsers or devices not supported (well)
- End User experience suffers from poor performance
- (Too) Simple passwords allowed



Owner: Product Owner
Found during: Dev, Test, Review, Usage

Impact/Risk (Immediate)
- Incidents – requiring costly resolution
- Missed business opportunities
- Dissatisfied users
- Public image suffers
- Loss of user productivity
- Security breach

Technical Debt

Owner: Architect (*
Found during: Dev, Code QA, Review

- Hard coded styles in web pages
- Low (to no) test coverage
- Complex, deeply nested, large, hard to understand code units
- Meaningless or even confusing variable names
- High degree of coupling
- Low quality documentation
- Magic numbers (hard coded values) in program code
- Use of deprecated òr unsupported technologies
- Manual steps in CI/CD
- Frequent substantial redesign of architecture/ platform / tech stack
- Libraries with security vulnerabilities

Impact/Risk (Longer Term)

- Changes increasingly become harder (lengthy, costly, risky) – low agility
- Hard to keep/find & motivate technical staff
- Drop in Team Productivity (velocity)
- Production incidents
- Increased Vulnerability

Operational (Ops) Debt

Ops Debt

Owner: Product Owner
Found during: Dev, Review, Ops, Usage

- Incomplete working instructions, checklists and operating guidelines
- Lack of SLA objectives and KPI definitions
- No regular “fire drill” (to test recovery, fail over, re-provisioning, …)
- No log file [management]
- No monitoring (on critical user experience or business metrics)
- No purging of temporary files or outdated data
- No backups taken (or recovery tested)
- Expiring certificates
- No fail-over done to standby environment when an incident happens

Impact/Risk (Short Term)
- High OPEX
- High number of incidents
- Slow response to incidents
- Unclear communication regarding incident status
- Incidents only handled when explicitly submitted (no auto detection)
- Loss of performance and availability
- Loss of data

Team & Process Debt

Team Process Debt

Owner: Scrum Master (*
Found during:
Retrospective, Audit, Refine, Dev, Review

- Incomplete *definition of ready* or DoR not applied
- No glossary of business terminology
- No clarity on business objectives, stakeholders
- Peer Review not [thoroughly] performed
- Lack of ownership from the team of the product[s]
- Dependency on individuals regarding specific components or tasks
- No automated functional [regression] testing
- Limited automation (in build, code QA, delivery/deployment)
- No coding standards (applied)
- Incomplete intake for products to go to Production (and be put under Ops)
- No onboarding instructions for new team members
- No register of design decisions
- No Continuous Improvement cycle based on periodic evaluation
- Frequent and repeated discussions (to realign, refocus)
- Culture in which pointing out deficiencies in someone else's work is not done

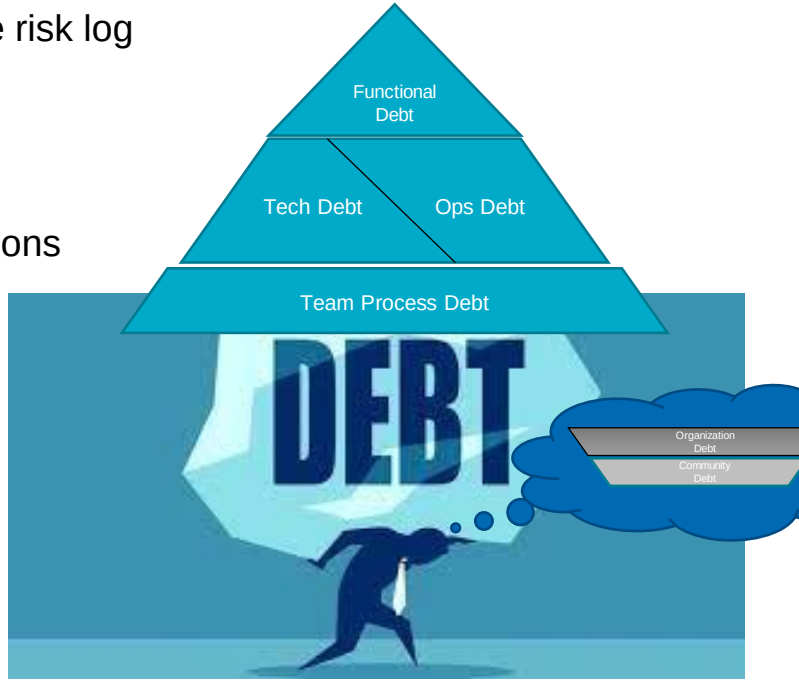
Impact/Risk (Mid Term)
- Accelerated build-up of all debt
- Loss of productivity
- Loss of synergy
- Loss of work pride and joy
- High impact of team changes

Moral Debt

- Organization
  - The team benefits from the organization and all it has in place:
    - reusable components, automation scripts, frameworks, platforms, infrastructure, standards, ways of working, tool instructions
  - The team should give back to the organization by improving existing artefacts and creating new ones; the organization should grow as a result of the growth in the team
- Community
  - The team makes good use of community resources: StackOverflow, blog articles, open source tools, libraries and frameworks
  - In order for the community to thrive, the team should make community contributions in return
    - Kudos, Enhancement Requests & Bug Reports, articles on using resources (why, how), actual Pull Requests for code

What to do about debt?

- Identify debt during
  - refinement, review, test, Ops intake, retrospective, audit, on boarding, periodic evaluation (LCM)
  - analysis, development, production usage
- Make debt explicit and visible – in a debt register and the risk log
  - what and where
  - severity, risk and impact (running cost!)
  - resolution: how and effort
- Discuss debt (risks, running cost of not fixing) & plan actions
  - in every sprint planning
  - in every steering committee session
  - …
- Continuously work on reducing debt – in small steps
  - boy scout principle (improve everything you touch)
  - explicitly set sprint budget aside
  - Debt status should be a Team KPI
- Focus on a root cause: Team Process Debt



The Debt Mountain