Technical Debt: Some Findings and a Standard

Dr. Bill Curtis, Executive Director
Cyber Resilience Summit June 6, 2017
The Technical Debt Metaphor

**Technical Debt** — the future cost of defects remaining in code at release, a component of the cost of ownership

- **Business Risk**
  - **Opportunity cost** — benefits that could have been achieved had resources been put on new capability rather than retiring technical debt
  - **Liability from debt** — business costs related to outages, breaches, corrupted data, etc.

- **Technical Debt**
  - **Interest on the debt** — continuing IT costs attributable to the violations causing technical debt, i.e., higher maintenance costs, greater resource usage, etc.
  - **Principal borrowed** — cost of fixing problems remaining in the code after release that must be remediated

- **Structural quality problems in production code**

Application Characteristics

- **Robustness** — stability, resilience, and recovery from operational interruptions
- **Performance** — responsiveness and efficient use of resources
- **Security** — ability to prevent unauthorized intrusions
- **Transferability** — understandability of an application by other teams
- **Changeability** — ease of modifying source code
- **Total Quality Index** — aggregate score of the 5 quality characteristics above
- **Size** — lines of code, Automated Function Points

Demographic Characteristics

- **Language/Technology**
- **Industry Sector**
- **Source and Shore**
- **CMMI Level**
- **Development Method**
- **Age & Team size**
Technical Debt by Quality Characteristic

- 70% of Technical Debt is in IT Cost (Transferability, Changeability)
- 30% of Technical Debt is in Business Risk (Robustness, Performance, Security)
- Health Factor proportions are mostly consistent across technologies

Agile/Waterfall Hybrid Best

Robustness

Performance

Security

Changeability

Transferability

All F-tests significant
$df = 4, 215; \quad p \leq .02$

Agile $n = 57$
Hybrid $n = 46$
No method $n = 21$
Other $n = 36$
Waterfall $n = 60$
Assumption: Productivity is a stable number

Reality: Productivity is a monotonically decreasing function of releases

Unless you take action !!!
Carry-forward Rework = Technical Debt

Release N
- Develop N
- Rework N

Release N+1
- Develop N+1
- Rework N+1
- Rework N

Release N+2
- Develop N+2
- Rework N+2
- Rework N+1
- Rework N

Unfixed defects release N
Unfixed defects release N+1
Unfixed defects release N

No Function Point credit for rework
Charge rework effort against the release where defect was injected
Reducing Cost of Production Outages

Large international investment bank

![Graph showing the relationship between Total Quality Index and Production Incidents, with three boxes highlighting different annual losses: +$2,000,000, $50,000, and $5,000.](graph.png)
Calculation of Tech Debt for a Quality Measure

AUTOMATED TECHNICAL DEBT MEASURE

∑ all patterns*

PATTERN REMEDIATION EFFORT

∑ all occurrences

OCCURRENCE REMEDIATION EFFORT

∑ for each occurrence

OCCURRENCE RAW REMEDIATION EFFORT

(adjacent on survey results)

ADJUSTMENT FACTOR

OCCURRENCE TECHNOLOGICAL DIVERSITY

OCCURRENCE COMPLEXITY OVERHEAD AVERAGE

OCCURRENCE EXPOSURE OVERHEAD AVERAGE

OCCURRENCE SHARING OPPORTUNITY AVERAGE

OCCURRENCE GAP SIZE
CISQ Technical Debt Measure

Correlate of effort to fix structural quality problems that must be addressed

Large component of cost of quality
CISQ specifies how to measure four quality characteristics and Automated Function Points – both at the unit level and the whole system level.

ANY ORGANIZATION CAN MEASURE THEMSELVES AGAINST CISQ SPECIFICATIONS.