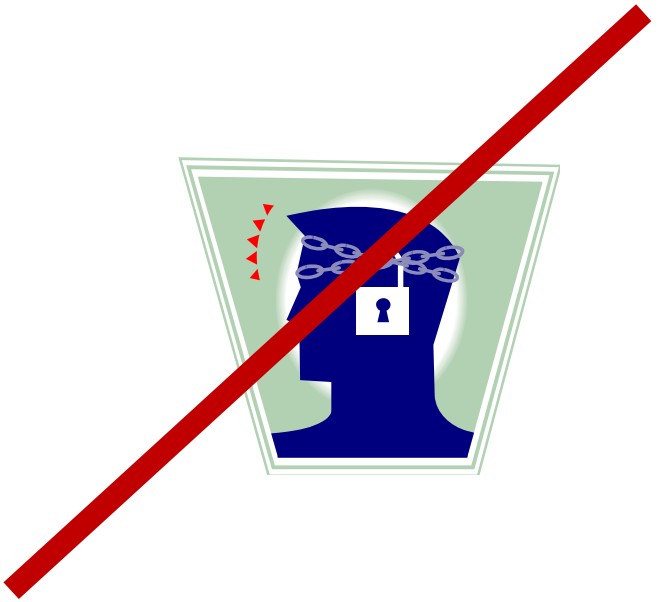
Harnessing Software Debt

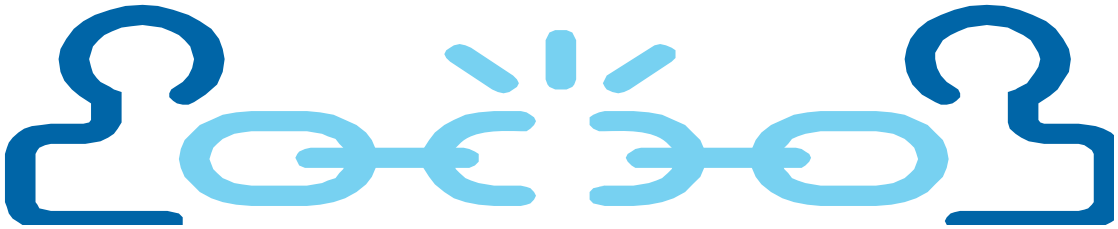
Cheryl McIntyre
Corporate Director of Software Engineering
Lockheed Martin

July 14, 2011

Today's Topics

- Examples of SW debt
- Measuring SW debt
- Managing SW debt





Software Debt Examples

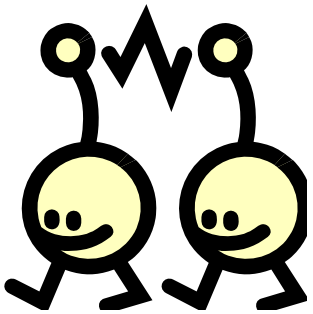
- Technical Debt
- Requirements Debt
- Decision Debt

What is Technical Debt?

- Two Aspects of Technical Debt
  - Time borrowed through a design or coding shortcut with the intention of paying back the debt as soon as possible, but…
    - Too many projects only pay back the interest on the debt in the form of bug fixes

  - Software Decay
    - Implicit form of technical debt that accrues unless explicit effort is made to prevent it

Examples of Technical Debt



- Design
  - Inconsistent approaches
  - Poor choice of components/frameworks
  - Hindsight design
- Code
  - Duplicate code
  - Overly complex modules
  - Standards violations
  - Lack of documentation
  - Style drift or clash
- Test
  - Incomplete coverage
  - Hard to maintain tests
  - Excessive tests

Today's tools make it easy to identify and measure many of these forms of technical debt.

Examples of Requirements Debt

TBDs and TBRs late in lifecycle

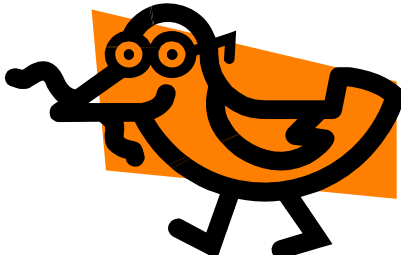
Missing TBDs and TBRs

Overly detailed requirements

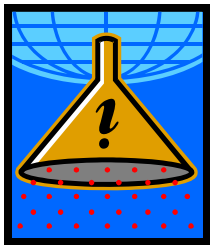
Obsolete requirements

Decision Debt

Decisions made too late



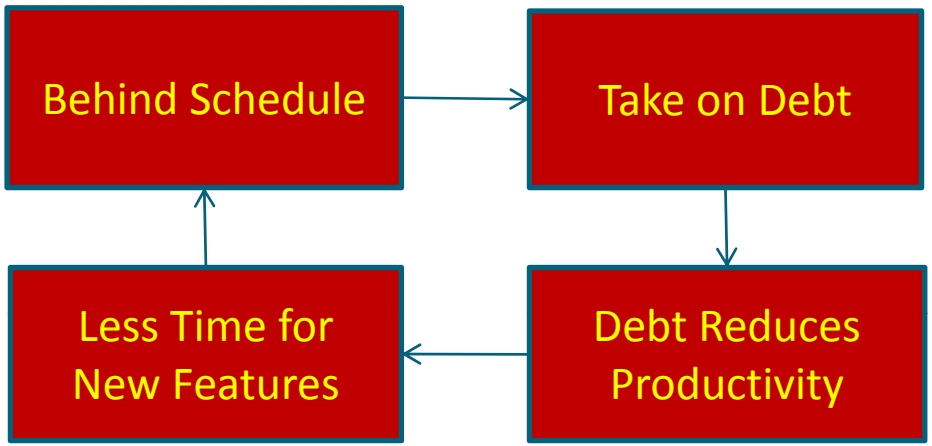
Decisions made too early

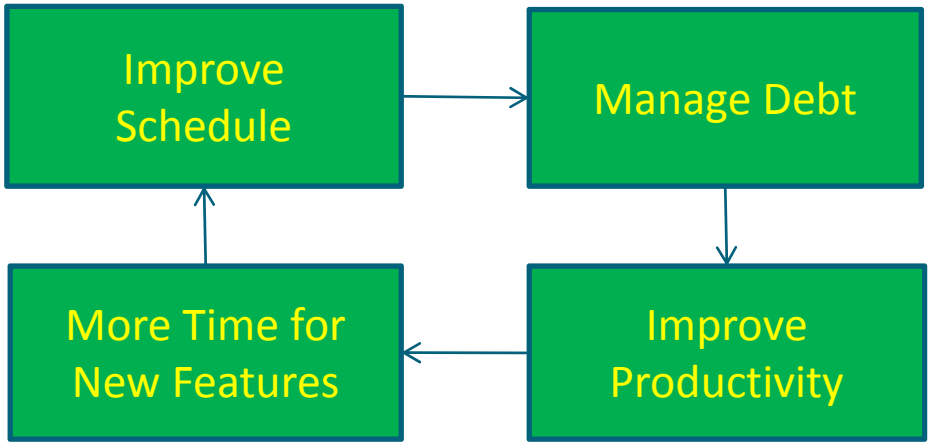


Decisions based on inadequate information

What is the level of confidence that you have in your decisions?

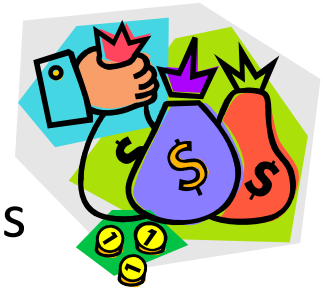
Debt Options





Why Do We Care About SW Debt?

- Measuring and controlling our SW debt can give us a competitive advantage in life cycle costs
  - Lower cost to develop
    - Code with less SW debt will have fewer defects and easier to fix defects
  - Lower life cycle cost
    - SW debt has a significant impact on the long term maintenance costs



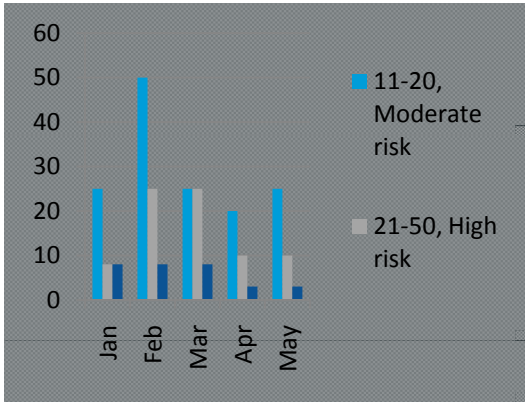
Life cycle affordability is a critical concern for our customers.

Measure and Manage– Technical Debt

- Technical debt from static analysis tools
  - Measure trends
  - Manage by setting thresholds for action and by identifying areas for developer attention

- Test coverage debt
  - Measure test coverage throughout the lifecycle
  - Manage by adding tests where needed to meet coverage goals or reduce risk

Module Complexity



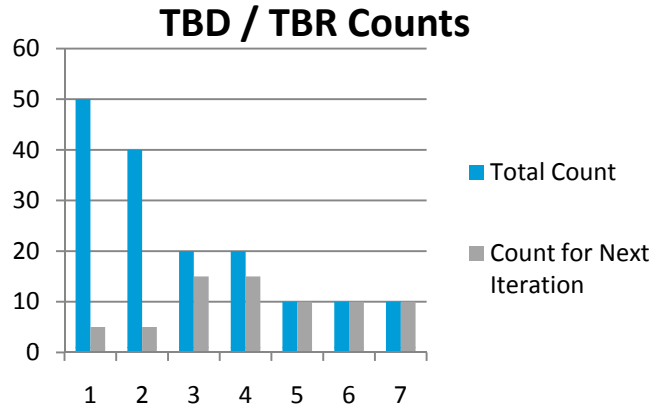
Test Coverage for Iteration N

Package Name	# Classes	Line Coverage	Branch Coverage
abcd	55	75%	64%
xyz	55	52%	43%
abqrs	3	0%	0%
vxxy	13	90%	75%
abcdefg	10	86%	88%

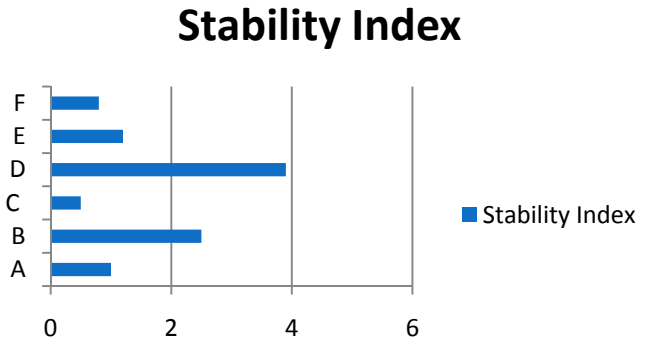
Controlling technical debt is essential for long-term success on Agile programs.

Measure and Manage– Requirements Debt

- TBDs and TBRs
  - Measure total count and count for requirements scheduled for implementation in the next couple of iterations
  - Manage to reduce open TBDs and TBRs in upcoming iterations



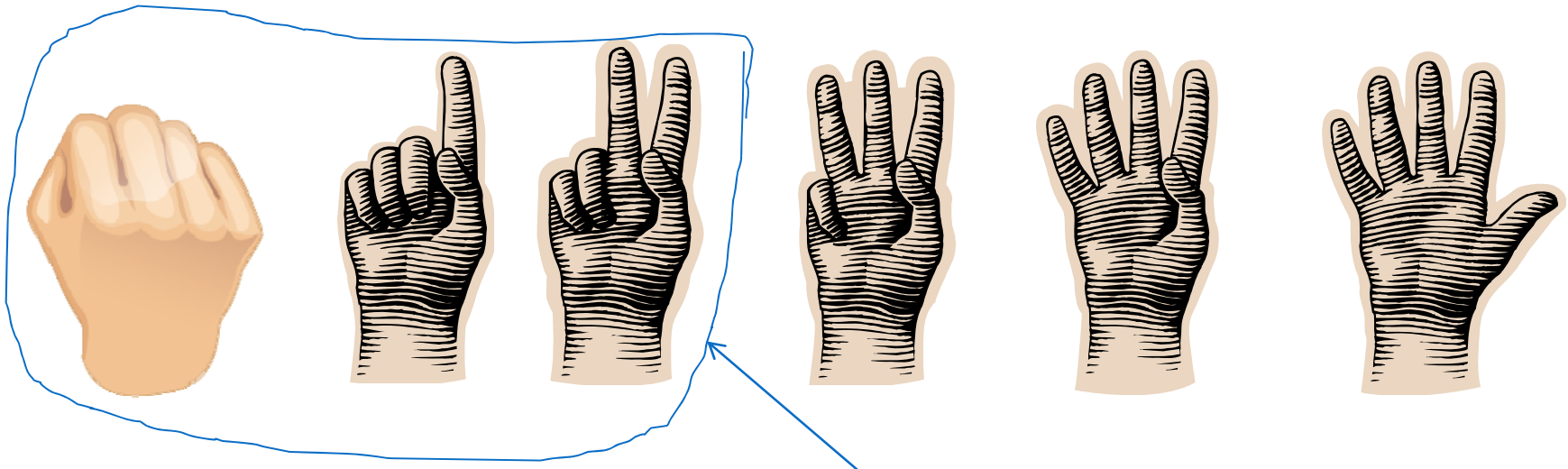
- Requirements Stability Index
  - Measure average complexity by component
  - Manage by using isolation patterns in areas with most instability



Respect software's limitations while leveraging its power.

Measure and Take Action – Decision Debt

- Decision confidence
  - Measure team confidence in each decision
  - Manage by reducing risk for low confidence decisions



Elicit the wisdom of the team.

Some Key Points for Managing SW Debt

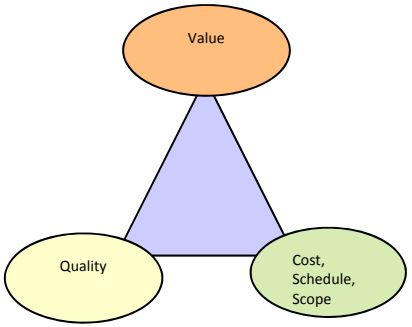
- Keep debt visible

- Distinguish between good debt and bad debt
  - Short term / long term
  - Focused / scattered
  - Intentional / unintentional

- Have a repayment plan

Final Advice

- SW debt has both management and technical consequences.
- Cost, schedule and technical debt should be tracked and managed together.
- Decisions based on SW debt need to be revisited throughout the lifecycle.
- Managing SW debt improves developer morale.



As an industry, we need to focus on measuring and managing SW debt.


QUESTIONS
&
ANSWERS