Workshop #7
What Does Technical Debt Mean at a System Level?

Bob Epps/Garry Roedler
Lockheed Martin Corporation
PSM User's Group Conference
July 14, 2011

Participants

- Mike Bandor
- Alejandro Bianchi
- Bill Curtis
- Mike Denny
- Bob Epps
- John Ertischweiger
- Cheryl McIntyre
- John Murdoch
- Alain Picard
- Garry Roedler
- Jim Stubbe

Technical Debt Workshop for Systems Engineering

Agenda

- Management of Technical Debt- Steve McConnell
- Technical Debt Observations- Jim Highsmith
- Types of Debt- Chris Sterling
- Break
- Workshop Exercise # 1-Identifying Technical Debt
- Management of Architectural Debt- Ipek Ozkapa
- Workshop Exercise # 2- Architecture & Technical Debt
- Workshop Exercise # 3- System Design & Technical Debt
- Workshop Summary/Action Plan

Technical Debt

"Management of Technical Debt", Steve McConnell

Summary of Kinds of Debt

Non Debt

Features backlog, deferred features, cut features, and so on. Not all incomplete work is debt. These are not debt because they do not require interest payments

Debt

I**. Unintentional Debt**. Debt incurred unintentionally due to low quality

II. **Intentional Debt**. Debt incurred intentionally

II.A Short-Term Debt. Short Term Debt, usually incurred reactively, for tactical reasons

II.A.1 Focused Short Term Debt. Individually identifiable shortcuts(like a car loan)

II.A.2 Unfocused Short-Term Debt. Numerous tiny shortcuts(like a credit card)

II.B Long-Term Debt. Long-term debt, usually incurred proactively, for strategic reasons

Types of Debt

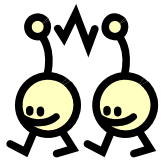
"Managing Software Debt: Building for Inevitable Change", Chris Sterling

- Technical Debt
  - These are activities that a team or team members choose not to do well now and will impede future development if left undone
- Quality Debt
  - There is a diminishing ability to verify the functional and technical quality of software
- Configuration Management Debt
  - Integration and release management becomes more risky, complex and error-prone
- Design Debt
  - The cost of adding features is increasing toward the point where it is more than the cost of writing from scratch.
- Platform Debt
  - The availability of people to work on software changes is becoming limited or cost-prohibitive.

Relationship to SW Technical Debt?

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.

- Do these examples apply to Systems?
  - All design items here are valid to systems
  - Although code does not apply, the sub-bullets apply if it were for requirements
  - Test items apply, but could be augmented to V&V
- Are there other sources of Technical Debt?
  - Could go through the system life cycle and determine where shortcuts are often taken

Relationship to SW Requirements Debt?

Examples of Requirements Debt

TBDs and TBRs late in lifecycle

Missing TBDs and TBRs

Overly detailed requirements

Obsolete requirements

- Do these examples apply to Systems?
  - Each apply
  - TBDs/TBRs late drive unintended change
  - Missing TBDs/TBRs drive potential rework due to missed analysis
  - Detailed requirements cross the boundary into design
- Are there other examples of Requirements Debt?
  - Operational Concepts?
    - Incomplete
    -
  - Stakeholder/mission requirements?
    - Same as the requirements debt here, but different perspective
    - Can come to contractor already with technical debt

Relationship to SW Decision Debt?

Decision Debt

Decisions made too late

Decisions made too early

Decisions based on inadequate information

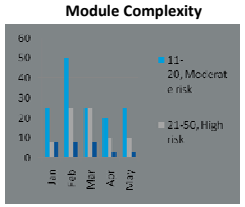
What is the level of confidence that you have in your decisions?

- Do these examples apply to Systems?
  - All apply
- What are impacts of decisions for:
  - Technology selection?
    - Too early – can provide obsolete/non-supportable solution
    - Too late – miss schedule
    - Inadequate info – choose wrong or immature technology
  - Manufacturability?
    -
    -
  - Sustainability?
    -
    -

How to Measure and Manage System Debt

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

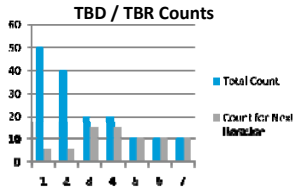
Test Coverage for Iteration N

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%

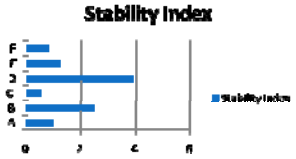
Controlling technical debt is essential for long-term success on Agile programs.

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.

- Do these measures apply for System Debt?
  - Fixes would be based on prioritization/severity
  - Technical reviews for architecture instead of Static Anl
  - Test coverage may apply
  - Requirements debt measures apply
- Other thoughts on measures
  - Confidence in architectural elements – could drive prototyping
  - Early validation of architecture
  - % of architecture integrated
  - Requirements validation debt (% requirements validated)
  - Requirements simulation debt (% requirements simulated)
  - Requirements verification debt (% requirements verified)

Technical Debt Workshop Action Plan

- White paper on Technical Debt applicability to Systems
  - Outline
  - Describe Technical Debt in Systems Engineering Vernacular
  - Identify sources and methods of measurement of Technical Debt within the Systems Engineering Life Cycle
    - System Requirements Analysis
    - System Architectural Design (all levels)
    - System Implementation
    - System Integration, Verification and Validation
    - System Transition (Deployment)
    - System Operations and Maintenance
  - Identify of implication of System Level Technical Debt to Software and Hardware Elements