STC 2015 Tutorial Part 3
Managing Technical Debt for Software and Systems Development Projects

IEEE Software Technology Conference
Long Beach, CA
October 12, 2015

Tutorial agenda

- Part 1: introductions and concepts of technical debt
- Part 2: Planning and organizing to manage technical debt
- Part 3: Managing technical debt
Part 3 topics

Part 3 topics include:
- The life cycle continuum of software development models
- Risk of and opportunities for managing technical debt within the continuum

The continuum of software development process models

- Highly Predictive: linear, one pass phases
- Incremental Predictive: incremental phases
- Iterative-Adaptive: iterative cycles
- Highly Adaptive: highly iterative cycles

see The Software Extension to the PMBOK® Guide Fifth Edition
Some factors in choosing a software development process

Factor to be considered include:

- Requirements clarity and volatility
- Level of stakeholder involvement
- Complexity and size of project and product
- Criticality of quality attributes
- Flexibility in allocation of schedule and budget
- Regulatory and policy constraints
  - e.g., safety certification or security accreditation

Selecting the development process

- Each development process encounters risks of accumulating technical debt
  - And provides opportunities to manage technical debt
- Selecting the wrong development process is a primary factor that contributes to accumulation of technical debt
  - and failure to mitigate technical debt
Risk and opportunity (1)

Risk is characterized by probability and cost
- Risk involves the probability that a negative event or situation will occur
  - For example, the probability of accumulating technical debt
- And the cost to mitigate a risk if it becomes a problem
  - The cost of paying technical debt that has been accumulated

Risk and opportunity (2)

- Opportunity is the probability that a positive event or situation will occur
  - The probability that technical debt will be detected before significant debt is incurred
  - The reduced cost of mitigating the debt when detected, as compared to letting it accumulate
  - Plus the opportunity to improve the development processes to reduce the probability and cost of future technical debt
The Software Development Process Continuum

- Highly Predictive: linear, one pass phases
- Incremental-Predictive: incremental phases
- Iterative-Adaptive: iterative cycles
- Highly Adaptive: highly iterative cycles

see The Software Extension to the PMBOK® Guide Fifth Edition

A Highly Predictive Development Model (HPDM)

- Elicit
- Analyze
- Architect
- Design
- Construct
- Integrate
- Test
- Accept

Stages:
- Elicit
- Analyze
- Architect
- Design
- Construct
- Integrate
- Test
- Accept

time
**HPDM Goals**

The goals for HPDMs are:

1. To initially develop requirements that are complete, correct, and stable
   And to develop a suitable software architecture
2. To accurately predict (initially) effort, cost, resources, skills, and technology needed to successfully complete the project
3. To complete each project phase, completely and correctly, before proceeding to the next phase

**Opportunities for managing HPDM technical debt**

Opportunities for managing HPDM technical debt include:

1. Establishing and maintaining traceability links
2. Maintaining baselines
3. Practicing version control
4. Conducting milestone reviews
5. Iteratively developing the software during the construction phase
The Vee model and technical debt

- The Vee model includes development of test plans as the work products are being developed
  - Which can identify risk factors for accumulation of technical debt
  - Because of issues that are incorrect, incomplete, inconsistent, or vague

A shortcoming of HPDMs: problems created first are detected last
Risk factors for HPDM technical debt

- Risk factors that increase the probability of incurring HPDM technical debt include:
  - Infrequent milestones
  - Ineffective milestone reviews
  - Infrequent stakeholder interactions
  - Lack of initial traceability
  - Failure to maintain traceability
  - Failure to maintain a project balance
  - Inadequate change control
  - End-of-project V&V

Traceability links and technical debt

Bi-directional traceability provides the following checklist of opportunities to detect and mitigate technical debt:

- Are all requirements present in the design?
- Are all design elements present in the code modules?
- Are integration and test plans traced to requirements and design?
- Other links?

Are there any missing elements?
Are there any extraneous elements?
Risk factors

- Risk factors for managing HPLC technical debt include:
  - Infrequent milestones
  - Infrequent stakeholder interactions
  - Lack of traceability
  - Failure to maintain a project balance
  - Inadequate change control
  - End-of-project V&V

Technical debt: maintaining a project balance

- Impact analysis is concerned with assessing the impact of proposed changes
  - In-scope changes versus out-of-scope changes

Product Scope & Quality attributes

- Schedule
- Budget
- Resources
- Technology

Failure to maintain a balance is a primary source of technical debt for HPDM projects
**Risks of HPDM change control**

- Insufficient impact analysis
- An ineffective CCB
- An un-empowered CCB
- Lagging CCB responses
- Failure to maintain a project balance
- Failure to verify and validate changes
- Failure to communicate changes to all affected parties
- Other risks?

**A Highly Predictive Life Cycle (HPDM)**

Diagram showing the process of HPDM with stages labeled Elicit, Analyze, Architect, Design, Construct, Integrate, Test, and Accept. The timeline is indicated with arrows showing the progression through these stages.
Reducing the risk of HPDM technical debt during software construction

- Iterative software construction can reduce the risk of accumulated technical debt for HPDM projects
- Iterative software construction involves:
  - Constructing the software in incremental stages
    - With incremental V&V
  - And with frequent demonstrations of progress (or lack thereof)
    - Internal demonstrations for the software developers
    - External demonstrations for the stakeholders

The risk of external demonstrations during software construction

- Stakeholders may request extensive changes based on demonstrations of the evolving software
  - This is a symptom of earlier accumulated technical debt
    - In the requirements and design phases
  - And may indicate that a HPDM is the wrong approach for the project
Factors that can reduce the risk of accumulating HPDM technical debt

- Stable, well-defined requirements
- Staged, cross-functional resources
- Familiar system
- Familiar customer
- Short duration project
- Adequate time, resources, skills, and technology
- Iterative software construction

---

The Software Development Process Continuum

- Highly Predictive: linear, one pass phases
- Incremental Predictive: incremental phases
- Iterative-Adaptive: iterative cycles
- Highly Adaptive: highly iterative cycles

see The Software Extension to the PMBOK® Guide Fifth Edition
An Incremental-Predictive Development Model (IPDM)

The IPDM

- Periodic demonstrations of system increments involve stakeholders
  - and may result in revisions to requirements
    - plus redesign and reallocation of feature sets
  - and reallocation of resources and schedule
- And can provide objective evidence of progress
  - Or lack thereof: accumulated technical debt
- Early deliveries of subset capabilities for evaluation by users are possible, if desired
IPLC Stakeholder Involvement

Increment demo intervals may be unequal

Some IPDM Capability Prioritization Criteria

- allocate the system requirements to increments of growing system subsets some of which are to be periodic delivered OR
- allocate the system requirements to architectural layers that incrementally result in a succession of layered virtual machines OR
- establish the architectural skeleton first and incrementally add capabilities OR
- establish interfaces to external components first OR
- incrementally incorporate components to be reused
How many feature sets are needed?

- The number of features per feature set and the duration of the incremental build phases for the feature sets may vary from feature set to feature set
  - But the durations of each incremental phase should not exceed one month

- Limiting the duration of each incremental phase to one month will determine the number of feature sets and incremental phases needed
  - And may require modifying the overall schedule and budget
  - Or dropping some of the lower priority requirements

Managing technical debt for IPDM projects

The opportunities and risk for managing IPDMs technical debt are similar to those for HPDMs

- With the added benefit of
  - Incremental verification and validation
  - Periodic stakeholder demonstrations of the evolving product
  - And the opportunity to make some periodic revisions to requirements and design
  - And to restructure the remaining feature sets
  - And delivery of subset capabilities to users for evaluation, if desired
Managing IPDM technical debt

IPDM technical debt can be successful managed when:

- Requirements can be mostly defined initially
- Some flexibility in managing prioritized requirements will be permitted
  - plus dynamic allocation of resources and schedule to increments
- Periodic V&V and stakeholder involvement provides feedback
  - as a basis for adding, deleting, and revising features
  - and to support management of technical debt

The Software Development Process Continuum

- Highly Predictive: linear, one pass phases
- Incremental-Predictive: incremental phases
- Iterative-Adaptive: iterative cycles
- Highly Adaptive: highly iterative cycles

see The Software Extension to the PMBOK® Guide Fifth Edition
An Iterative-Adaptive Development Model (IADM)

Internal Iterations by Adaptive Teams

Internal demonstrations of evolving capabilities may occur on a daily basis
Test-driven development (TDD)

- Internal demonstrations, test-driven development, and refactoring are three iterative approaches to controlling technical debt
- Internal demonstrations provide evidence of progress
- TDD involves writing test scenarios and test cases before writing code
  - May include both static and dynamic tests
- Refactoring involves restructuring the software without altering existing behavior
  - Why not alter behavior?

Adaptive Software Development

- Technical debt can be successfully managed when pursuing an adaptive approach by:
  - developing an initial understanding of the nature of the desired product to be delivered,
  - adopting an architectural style or establish a shared design metaphor
  - determining the constraints on schedule, budget, resources, and technology that must be observed
  - developing a continuous, ongoing relationship with a knowledgeable user representative
  - periodically demonstrating the evolving software
  - Concurrently evolving the requirements and the software
AND BY observing the 10 attributes of agility

Iterative-Adaptive projects are “agile” to the extent that they observe the 10 attributes of agility

The 10 attributes of agility are included at the end of these slides

Controlling technical debt by tracking IADM progress

- Burndown charts and velocity charts can be used to track progress and control technical debt for IADM projects
A Burndown Chart for an IALC Iteration Cycle

- Planned tasks remaining
- Actual tasks remaining

Tracking Progress Using a Velocity Chart

- Series2

Iteration Cycle in Work-Days

Number of Remaining Tasks

0  3  6  9  12  15

0  5  10  15  20  25  30
Carrying Tasks Forward

- Retrospective meeting:
  - Why didn’t we complete everything?
- Planning meeting:
  - How are we going to handle the carry-forward?
    - While controlling technical debt

  - The retrospective meeting and planning meeting can be combined
    - unless there is a time lag between iterations
    - as in staged procurement or maintenance upgrades

Handling carry-forward of IADM tasks

Persistent carry-forward of tasks accumulates technical debt

- Desirable options for paying IADM technical debt
  - Use occasional iteration cycles to clear accumulated technical debt
  - Re-scope the number of tasks that can be completed during each time-boxed iteration cycle
  - Deliver fewer prioritized requirements
  - Extend the schedule
  - Add an IADM mentor to the team
  - Spin off some of the work to another team
Handling carry-forward of IALC tasks

Undesirable options for paying IALC technical debt

- Add more team members
- Replace team members (except in extreme cases)
- Reduce TTD, refactoring, and verification testing
- Reduce external demos and interactions with the customer
- Require overtime

The Software Development Process Continuum

- Highly Predictive: linear, one pass phases
- Incremental-Predictive: incremental phases
- Iterative-Adaptive: iterative cycles
- Highly Adaptive: highly iterative cycles

see The Software Extension to the PMBOK® Guide Fifth Edition
A Highly Adaptive Life Cycle

Attributes of HADM projects

- Customer controls the backlog of use cases or stories
  - and is continuously “in the loop” with the software developers
- Demonstrations of progress occur on a daily basis
  - Or multiple times per day
- The “development team” may be 1 or 2 persons
### Controlling technical debt for HADM projects

- There are no issues of technical debt because progress is continually demonstrated
  - And is under the control of the customer

---

### The Software Development Process Continuum

- **Highly Predictive**: linear, one pass phases
- **Incremental Predictive**: incremental phases
- **Iterative-Adaptive**: iterative cycles
- **Highly Adaptive**: highly iterative cycles

Goals for this tutorial

- To present the nature of and consequences of technical debt and
- To present techniques for anticipating, detecting, and mitigating technical debt and to
- Address your questions and concerns
- And hear your recommendations for improving the processes, methods, tools, and techniques for avoiding and managing technical debt
Tutorial agenda

- Part 1: Concepts of technical debt
- Part 2: Planning and organizing to manage technical debt
- Part 3: Managing technical debt

Part 1 topics

- Concepts of debt
- Accumulation of technical debt for software development
- Interest rates for technical debt
- Reasons for knowingly and unknowingly accumulating technical debt
- Passing it on
Session 2 topics

- Excuses for not planning
- Making a plan for the technical work (SwTMP)

Part 3 topics

- The life cycle continuum of software development models
- Opportunities and risks for managing technical debt across the continuum
Questions?
Comments?

10 Attributes of Agility for Iterative-Adaptive projects
Attributes of Agility for Adaptive SDLCs

1. Increments of working deliverable software are produced on a periodic basis
2. Adaptive iteration cycles are of the same duration (i.e., are “time boxed”) for a project but some cycles may be of longer or shorter duration by exception
   - durations of the time-boxed iteration cycles may be weekly, bi-weekly, or monthly but not more than monthly
3. Increments of working deliverable software are not necessarily produced by each iteration
   - increments and iterations are distinct
4. Requirements, design, and the software product emerge as the project evolves

Attributes of Agility (2)

5. A representative customer, customer’s representative, and/or knowledgeable user is involved on a continuing basis
   - involvement includes observing periodic demonstrations of working, deliverable software at the end of each iterative development cycle
   - In addition, a representative customer or customer’s representative accepts shared responsibility for further product development based on demonstrations of working deliverable software
     - and accept responsibility for managing the constraints on project scope (schedule, budget, and resources)
Notes

- The customer representative must commit to:
  a. Being continuously available
  b. Providing updates to the requirements based on periodic demonstrations of the incrementally evolving product that involve the customer and other key stakeholders
  c. Discussing proposed changes to the requirements with the software developers to assess technical risk and the impact of proposed changes on the product, schedule, and needed resources
  d. Accepting responsibility for maintain a balance among schedule, budget, and requirements to be implemented

Attributes of Agility (3)

6. Adaptive software development teams are small (i.e., 10 or fewer members) and are self-organizing
   o large projects include multiple small teams
7. Short daily standup meeting, retrospective meetings, and planning meetings are used
8. Each software development team is cross-functional
   o includes the combination of skills needed to accomplish the work activities
   o functional experts may be involved occasionally as needed
Attributes of Agility (4)

9. All members of each software development team are assigned to one project at a time

10. Overtime is not required
    - Except on rare occasions of limited durations
    - Software developers should be able to maintain a steady pace indefinitely
    - Without burnout
    - And without sacrificing their personal lives