Tutorial agenda

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

• Excuses for not planning
• Making a plan for the technical work to be done

Planning guidelines

- Technical plans should be developed initially, to the extent possible
  - Regardless of the development process to be used
- But a plan should not be based on unachievable constraints
  - Or unrealistic assumptions

Failure to plan, to the extent possible, is a primary cause of unknowing accumulation of technical debt
Excuses for not planning

- Not enough time to make a plan
- Inadequate planning skills
- Insufficient information
- Lack of organizational support
- “Why make a plan?; everything will change anyway”
- “I’m a doer, not a planner”
- “Trust me, I have it all right here in my head”
- Other excuses?

A well-known saying

“Failing to plan is planning to fail”
Session 2 topics

• Excuses for not planning
• Making a plan for the technical work to be done

Planning the technical work

- Systems engineers distinguish between the project plan and the technical plan
  - By developing a Systems Engineering Management Plan (SEMP) for the technical work to be done
  - Which is distinct from the system engineering project (or program) plan (SEPP)
    - Which covers issue of schedule, budget, resources, facilities, reporting, and so forth
1.0 Purpose of the document
2.0 Scope of the project
3.0 Technical planning and control
4.0 Systems engineering process
5.0 Transitioning critical technologies
6.0 Integration of the system
7.0 Integration of the system engineering effort
8.0 Applicable Documents

A SEMP checklist

- Are all of the technical challenges of the project addressed by the system engineering processes described in the SEMP?
- Does the SEMP describe the design processes and the design analysis steps required for an optimum design?
- [Does the SEMP account for the design constraints and design criteria to be satisfied]*

* added
A SEMP Checklist (2)

- Does the SEMP clearly identify any necessary supporting technical plans, such as a Verification Plan or an Integration Plan?
  - Does it define who, when, and how they will be written?

A SEMP Checklist (3)

- Does the SEMP spell out all the required technical staff and development teams?
  - Does it identify the technical roles to be performed by the system’s owner, project staff, stakeholders, and development team
    - [and needed skills and skill levels]
  - Does the SEMP cover the interfaces among the various development teams?
    - [and team member roles to be played]?
Planning the software technical work

- Software engineers would be well-advised to develop software engineering technical management plans (SwTMPs)
- The project manager develops and owns the software engineering project management plan (SwPMP)
- The lead software engineer (i.e., chief software architect) develops and owns the SwTMP
- The SwPMP and the SwTMP are cross-referenced and coordinated

A Key Point

- A division of responsibility between the project manager and the chief architect is essential
  - with close coordination and collaboration
A Software Technical Management Plan (SwTMP)

Title page

1.0 Purpose of the document
   1.1 revision history

2.0 Scope of the *product to be delivered*
   *prioritized features, behavioral properties, and quality attributes*

3.0 Technical planning and control
   *including supporting processes*

4.0 Software development process
   *see part 3 of this tutorial*

* Italicics added for emphasis

Some Supporting Processes for Software Projects

<table>
<thead>
<tr>
<th>Supporting Process</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Management</td>
<td>Change control; baseline management; product builds; product audits</td>
</tr>
<tr>
<td>Verification</td>
<td>Determining the degree to which work products satisfy the conditions placed on them by other work products and work processes</td>
</tr>
<tr>
<td>Validation</td>
<td>Determining the degree of fitness of work products for their intended use in their intended environments</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>Assuring conformance of work processes and work products to policies, plans, and procedures</td>
</tr>
<tr>
<td>Documentation</td>
<td>Preparation and updating of intermediate and deliverable work product documentation</td>
</tr>
<tr>
<td>Developer Training</td>
<td>Maintaining adequate and appropriate skills</td>
</tr>
<tr>
<td>User and Operator Training</td>
<td>Imparting skills needed to effectively use and operate systems</td>
</tr>
</tbody>
</table>

* Italics added for emphasis
Selecting the development process

Selecting the wrong development process is a primary factor that contributes to accumulation of and failure to mitigate technical debt.

Part 3 of this tutorial presents techniques for managing technical debt for different development process models.

A Software Technical Management Plan SwTMP (2)

5.0 Transitioning critical technologies
to include software to be reused and vendor/subcontract supplied components
and any integration plans for hardware and software

6.0 Integration and verification of the software
plus validation, acceptance, and delivery of the software

7.0 Integration of the software engineering effort

8.0 Applicable Documents
7.0 Integration of the software engineering effort

Integration of the software engineering effort is primarily concerned with establishing good communication among team members and among teams
- Because software development is a human-intensive endeavor
- Communication becomes more complex as products grow larger and more complex
  - Because more people having more diverse skills are needed

Sizing the development team(s)

A software development team should be limited to 10 or fewer members
- Because the number of bi-directional communication paths within a team of n members is \( P = \frac{n(n-1)}{2} \)

Examples:
- For 5 team members \( P = 10 \)
- For 10 team members \( P = 45 \)
- For 20 team members \( P = 190 \)

Inadequate communication is a primary cause of unknowing accumulation of technical debt
Integrating the team

- Team of 10 or fewer members can be largely self-organizing
- One person may play the role of project manager and software architect

What about larger projects?

Integrating the team (2)

- Medium to large projects
  (2 to 5 teams of 10 or fewer members each)
  and very large projects
  (projects of 50 or more software developers)
  must use traditional mechanisms of engineering project coordination and communication, including:
  - hierarchical decomposition
  - shared interface specifications
  - shared change control mechanisms
  - shared integration plans and activities
Coordinating the work (2)

- Each small development team can use an appropriate development process for the kind of work they are doing

Scaling of the SwTMP

- Small simple projects have small simple plans for the technical work to be done
- Large complex projects have large complex plans for the technical work to be done
- It is not a question of what to leave out of a SwTMP but how to scale it to be effective without being bureaucratic
Scaling of the SwTMP (2)

- For software projects, plans for the technical work to be done typically evolve as the work progresses
  - and as knowledge of the work to be done grows

Failure to make, evolve, and update plans for the technical work to be done is a major cause for accumulating technical debt

Session 2 topics

- Excuses for not planning
- Making a plan for the technical work to be done
Next

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