MANAGING FOR SOFTWARE RELIABILITY

Steven A. Stolper, Software Consultant
info@stolperconsult.com
Software Management affects Reliability

- Risk Management
- Software Estimation
- Development Practices*
- Dealing with the “Death March” (Crises)
Risk Management (Identification)

- Identify technical risk (mindfulness)
- Question until you understand
- Formulate strategy to mitigate
- Manage as a resource
- Examples:
  - New technology, New flight hardware, Remote engineering, Small staff, hardware arrives late in development, new development tools...
- Justify technology use
### Risk Management Continued (Mitigation)

#### Conjoint measurement:

<table>
<thead>
<tr>
<th>Risk area</th>
<th>Potential risk</th>
<th>Expectation of the risk (1 to 5)</th>
<th>Impact of the risk (1 to 5)</th>
<th>Severity of the risk (Expectation X Impact)</th>
<th>Mitigating strategies &amp; controls</th>
<th>Decision summary</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule / Budget</td>
<td>Parts arrive late from suppliers</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>Safety margin of 10% incorporated into lead time for critical components</td>
<td>Specifics of how we mitigate this risk go here</td>
<td>J. Smith</td>
</tr>
<tr>
<td>Schedule / Design / Technical</td>
<td>Staff turnover</td>
<td>3</td>
<td>4</td>
<td>12</td>
<td>Technical leads will maintain knowledge of team member's work. Each technical lead will have a deputy who has &quot;high-level knowledge&quot; and &quot;history&quot;</td>
<td>Specifics of how we mitigate this risk go here</td>
<td>A. Thomas</td>
</tr>
<tr>
<td>Technical</td>
<td>First use of RTOS and compiler tool-chain</td>
<td>3</td>
<td>4</td>
<td>12</td>
<td>Hire at least one developer with previous experience with toolchain. Contract with vendor for developers to attend classes. Perform benchmark testing.</td>
<td>Specifics of how we mitigate this risk go here</td>
<td>A. Thomas</td>
</tr>
</tbody>
</table>
Technical Debt (Unintentional)

- Insufficient up-front design (architecture)
- Not enforcing good implementation practices
  - Modularity
  - Sanity checks
  - Comments / documentation
- Not enforcing good development practices
  - Concise specifications / requirements
  - Design reviews
  - Code Reviews
  - Monitoring check-ins
  - Fixing compiler warnings
- Major headaches in QA/ATLO or IN FLIGHT!
- A MESS IS NOT TECHNICAL DEBT!!!
Why is Good Software Estimation Important?

- Avoid the “Death March”
  - Twelve-hour workdays and weekends
    - Mental fatigue
    - Dumb mistakes
    - Shortcuts
- Increases technical debt

Bugs!!!!! Bugs!!!!!
Software Estimation

- Scheduling:
  - Work Breakdown Structure (WBS)
    - Activities required to complete project
  - Estimate
    - Work required to complete activities
  - Schedule
    - Ordering of activities and assignment of resources
Software Estimation Basics

- A target is not an estimate!
- What is a good estimate?
  - For commercial software, within 25% of actual, 75% of the time (Conte, Dunsmore, Shen)
  - Some organizations have achieved accuracy within 10%, but don’t hold your breath.
  - Are you 90% confident? (best->worst)
The Cone of Uncertainty

Please see renditions of the “Cone of Uncertainty” widely available on the Internet.
Estimating (cont)

- The anchoring effect
  - Avoid off-the-cuff estimates
  - Spend time on your estimates
  - “A date for a date”
- Be willing to overestimate
  - If we are good, should we be over as often as we are under?
- When to commit?
  - Initial high-level architecture (What are the pieces?)
  - The “Last Responsible Moment”
- Manage margin as a resource (to hide or not?)
Estimating Methodology

- Use more than one technique
  - Expert judgment
    - i.e.: analogy, PERT
  - Group estimates
    - i.e.: Planning poker
  - Scenario-based
  - Computing/Decomposition
    - i.e.: features, function points
  - Historical analogy
    - measurements from past projects
    - Actual, not scheduled!!! 😊
Crises: The “Death March”

I love deadlines.
I love the whoooshing sound they make as they fly by.

Douglas Adams
What to do?

- No magic bullet
- Create a new schedule with accurate estimates
  - We are looking for reality, not happiness
- Show it to management
  - They won’t be happy
- Discuss options
  - Move dates
  - De-scope/defer Features
    - Beware the stair-step effect
  - Add resources*
  - Retire to Hawaii
Summary

- Good software management avoids the “Death March”
- The “Death March” increases technical debt and undermines software quality/reliability
- Work to avoid the “Death March”
  - Identify and manage risk
  - Be mindful of technical debt
  - Improve your software estimates
Additional Reading

- “Software Estimation: Demystifying the Black Art”, Steve McConnell
- “Ship it! A Practical Guide to Successful Software Projects”, Jared Richardson and William Gwaltney