Overview

• GAO Study - Large scale project assessment
• Standish Group CHAOS Study
• Difficulties specific to the aerospace industry
• Agile methods vs heavy waterfall methods
• Where do we go from here
Disclaimer

- Will focus on projects with NASA as customer
- NASA is not the bad guy
  - NASA has done an uncountable number good things
  - Software engineering is ‘just hard’
In February 2010, the United States Government Accountability Office (GAO) did an assessment on selected large scale NASA projects.

- GAO assessed 19 NASA projects with a combined life-cycle cost of more than $66 billion.

Findings were that most missions were over budget, and behind schedule.
- Of those 19 projects, 4 are still in the formulation phase where cost and schedule baselines have yet to be established, and 5 just entered the implementation phase in fiscal year 2009 and therefore do not have any cost and schedule growth.
9 of the 10 projects that have been in the implementation phase for several years experienced cost growth ranging from 8 to 68 percent, and launch delays of 8 to 33 months, in the past 3 years.

These 10 projects had average development cost growth of almost $121.1 million—or 18.7 percent—and schedule growth of 15 months, and a total increase in development cost of over $1.2 billion, with over half of this total—or $706.6 million—occurring in the last year.
<table>
<thead>
<tr>
<th>Project</th>
<th>Critical technology maturity</th>
<th>Complexity of heritage technology</th>
<th>Design stability</th>
<th>Contractor performance</th>
<th>Development partner performance</th>
<th>Funding issues</th>
<th>Launch manifest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aquarius</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Glory</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRAIL</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Herschel</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Juno</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JWST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Kepler</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>LRO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MMS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>RBSP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOFIA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WISE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ares I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>GPM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>LDCM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Orion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: GAO analysis of NASA project data.
Example - GLORY

Initial baseline cost at $168.9

Cost went way over, and was re-baselined at $259.1

Latest cost Oct. 2009 at $296.1 which is a 14.3% increase since the re-baseline, and a total increase of 75%!
Cobb’s Paradox

• “We know why projects fail, we know how to prevent their failure -- so why do they still fail?”
  - Martin Cobb Treasury Board of Canada Secretariat, Ottawa, Canada

• Standish Group CHAOS Study

• 10 Reasons Projects are late, and/or over budget relating to software development
Success Criteria

1. User Involvement -19
2. Executive Management Support -16
3. Clear Statement of Requirements -15
4. Proper Planning -11
5. Realistic Expectations -10
6. Smaller Project Milestones -9
7. Competent Staff - 8
8. Ownership - 6
9. Clear Vision & Objectives - 3
10. Hard-Working, Focused Staff - 3
TOTAL 100
User Involvement

• Typically, user involvement consists of:
  • engineers, operators, and scientists who are located in separate facilities, separate states, or even separate countries

• Long distance can hamper communication paths and thus iterative user involvement
  • Often software developers never even meet some of the users

• Politics often plays a role in user involvement, and not for the better
  • Mars Climate Orbiter mission failed due to famous unit conversion error
Mars Climate Orbiter 1998
- Failed due to unit conversion error
  - Imperial units vs metric
- Caused spacecraft to enter Mars orbit at 57 km instead of the planned 140-150 km orbit
- Drag likely destroyed the vehicle
- Communication errors during testing (between contractor) likely a root cause of this failure
Executive Support & Ownership

- Failure consequences are unacceptable
  - Budgets are high, tax dollars are at stake
  - Projects failing badly have to explain themselves before congress

- Does the project team have a stake?
  - Often there are no direct incentives for success by individual developers
  - Contractors may have bonuses, but largely, the business if funded by government money and there are no ‘profits’ to share if success is ample

- Who is the boss?
  - Isn’t always clear who is in charge sine NASA provides oversight and teams also have dedicated executives
Clear Statement of Requirements

• NASA is big on requirements
  • Insists on formally documented, tracked, and verified requirements

• But requirements are not flexible, or iterative
  • Requirements are viewed by NASA as being solidified up front via a waterfall process definition
  • Does not anticipate and/or allow for iterative requirements
  • Makes it difficult for a contracted organization to implement their own Agile software process beneath
Proper Planning & Milestones

- Schedules are ‘success oriented’
  - But often failure to meet schedule date is unacceptable
  - Launch date may be physically unable to move due to planetary alignments
  - Launch date may be extremely costly to move due to schedules of launch pad activities and launch vehicles

- Does not follow 80/20 rule
  - “20% of a project's features will provide 80% of user benefits”
  - Missions are built on meeting science (or other mission) objectives not on benefits to the user which may be ‘negotiable’. Everything is decided up-front
  - Fixes or updates or improvements post launch are often unacceptable or impossible
Testing Schedule Compromised

- Due to inability for milestones to slip, there are bound to be consequences
- Testing cut short
  - Integration tests merged with system tests
  - System tests merged with acceptance tests
- Impact of adding requirements understated, and impact of removing requirements is under analyzed
Code Bases

- Facilities usually develop software for more than one mission
- Missions have overlap in code, but not necessarily in process definition or rigor
- Changing the code for mission A may also change it for mission B
  - Different missions have different testing, documenting, and verification processes
  - Leads to interesting and sometimes costly configuration management concerns
Conclusion

• So we know **that** projects are over budget and behind schedule
• And we know **why** projects are over budget and behind schedule
• This is not a unique problem to aerospace software
  • Most all software projects have similar concerns, perhaps not as strict
What would NASA do?

- There are two extremes to the pendulum
  - More strict bloated process, more meticulous and verbose documentation
  - Streamlined (Agile) processes, only documenting that which provides added benefit

- Over the past decade, NASA has been leaning towards the former solution, not the latter
  - NASA has adopted CMMI to measure their process model
  - In itself, this does not imply a heavier process, but can add to cumbersome oversight when CMMI is attempted to be flowed down to other organizations who might do things differently
Agile Methods

- Agile software development focuses on iterative development
- Uses constant small measurements to evaluate and adapt
- Teams focus on meeting functionality requirements based on user stories and iterative user involvement
- Agile doesn’t mean less process, it means more focused process
Agile Methods in Aerospace

- Since NASA executes projects in a waterfall process, working Agile methods beneath is difficult
  - NPR 7150.2 - standards NASA levies on all contracted organizations
  - Defines much of the processes, documentations, and plans that NASA requires for all work on a given project
  - Requirements are due and finalized up front, prior to completing the design, much less actual implementation of the design
  - Specific documentation is often required
    - Keeping user stories on note cards may not be acceptable and would have to be done in addition to other redundant efforts of tracking work
Added Risk?

- NASA may be reluctant to adopting more Agile methods due to a perception of added risk
- Peter Norvig quotes NASA administrator Don Goldin as saying:

  We’ve got to do the better, faster, cheaper. These space missions cost too much. It’d be better to run more missions and some of them would fail but overall we’d still get more done for the same amount of money.

- Coders at Work (by Peter Seibel)
Where are we going from here?

• Added complexity and weight to the development process does not appear to be helping

• As NASA begins to work with more commercial organizations to develop software, cost savings could be found by allowing them to develop the software under their own methods
  • A delicate balance between managing the outcome of the software with managing the development of the software must be achieved
  • Always keeping in mind, there is ‘No Silver Bullet’ (Fred Brooks) that will suddenly make software engineering in aerospace an ‘easy’ task