Improving Performance on Military Software Projects

Donald M. Beckett

Quantitative Software Management, Inc.

2000 Corporate Ridge, Suite 900
Mclean, VA 22102
Tel: 703 790-0055, Fax 703 749-3795
Email: info@qsm.com
Web: www.qsm.com

8/2/2012

Game Theory

- **Zero sum game**
  - I win you lose or
  - You win I lose
- **Non-zero sum game**
  - I win you win or
  - I lose you lose
- How can the government and systems integrators move from a zero sum game to a non-zero sum game where both sides succeed?

A Modest Proposal

- Government software projects and programs with a large software component can be completed at less cost and with higher quality *if*

Both the government and the systems integrators make some adjustments

Just what are these adjustments?

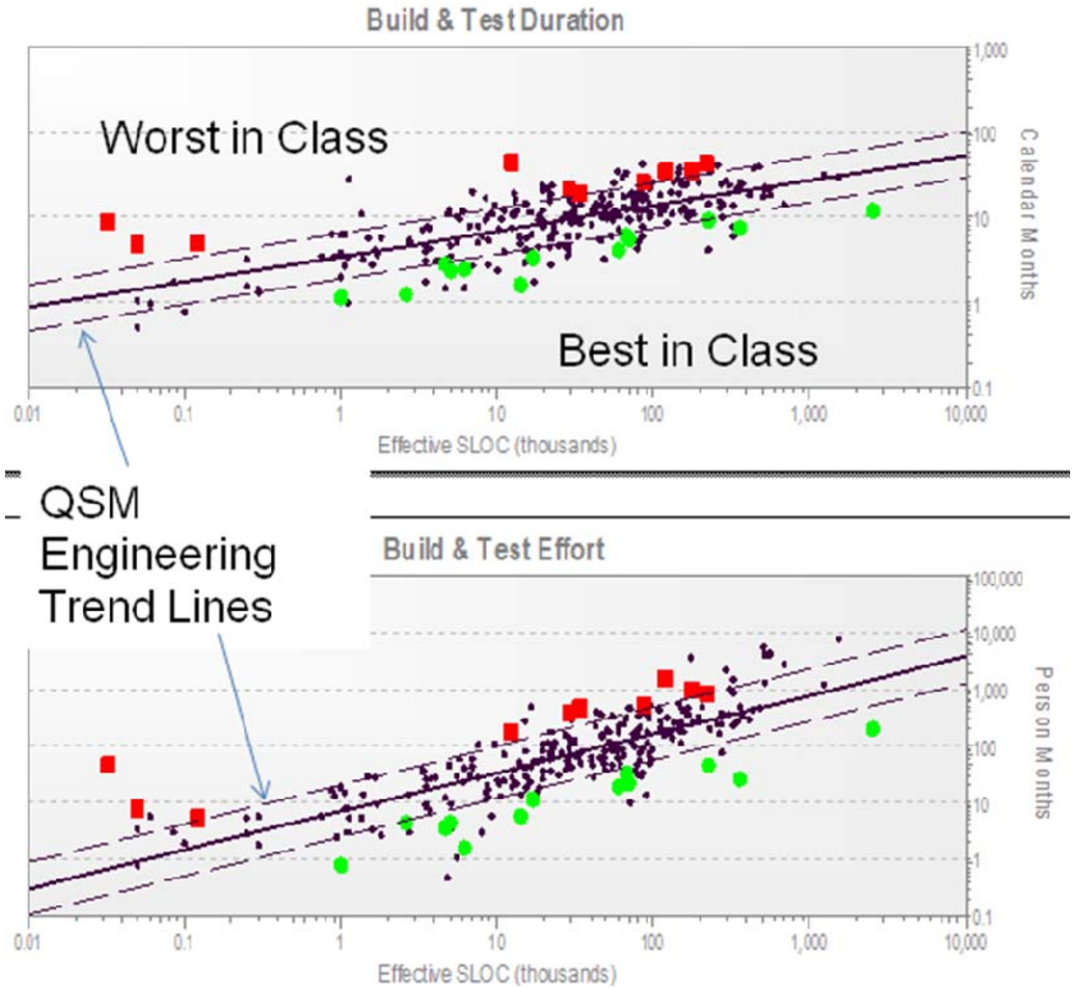
Outline

- Engineering systems Best-in-Class Worst-in-Class projects: what they teach about
  - Schedule
  - Effort
  - Staffing
  - Quality
- Military vs. non-military projects
- Dynamic tension
  - Shared and unshared goals
  - Potential gottcha's
  - Embracing uncertainty
- A way forward

Best-in-Class Worst-in-Class Engineering Projects

- Analyzed 276 completed Engineering domain projects
  - Engineering domain includes Command & Control, Telecomm, Scientific, and Systems Software
  - Does not include Business IT, ERP implementations, Real Time, or Microcode
- Best-in-Class projects were 1σ better than average; Worst-in-Class 1σ worse for both time to market and cost/effort

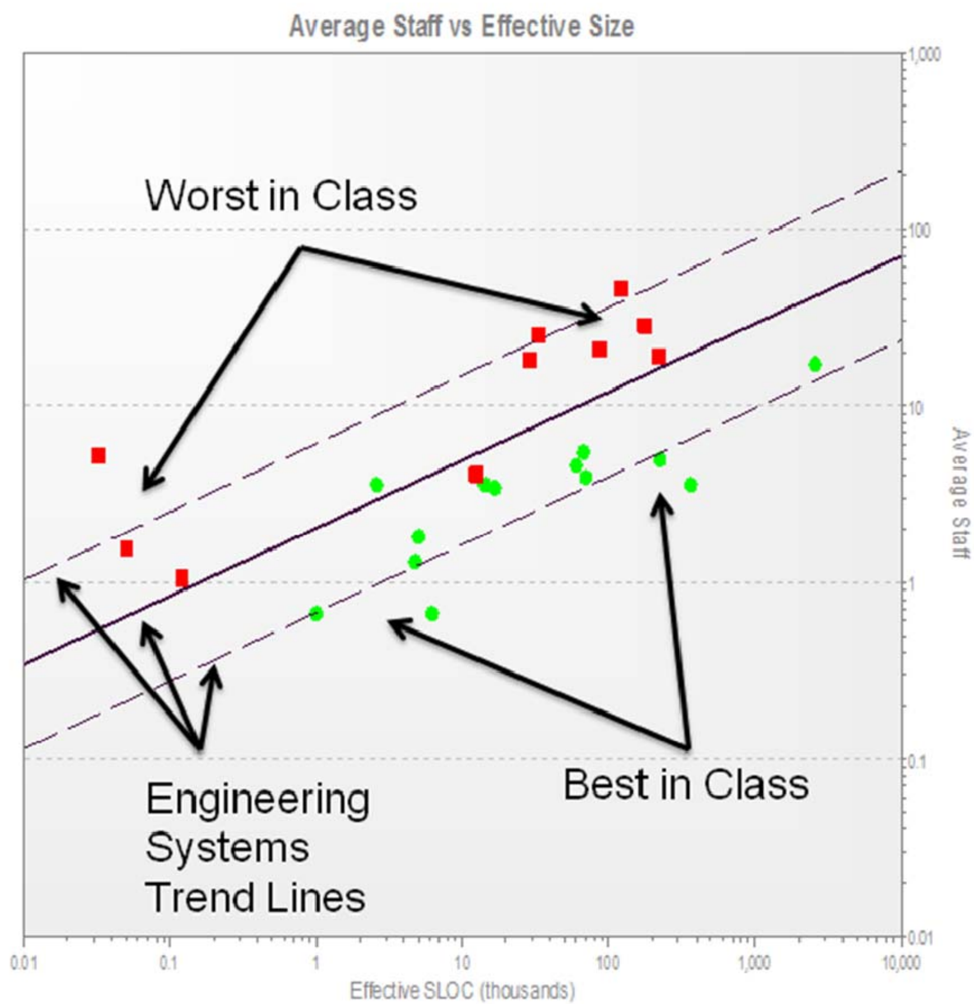
Best-in-Class Worst-in-Class Schedule and Effort



Best in Class project spread over size spectrum
Worst in Class concentrated in projects larger than 10k lines of code

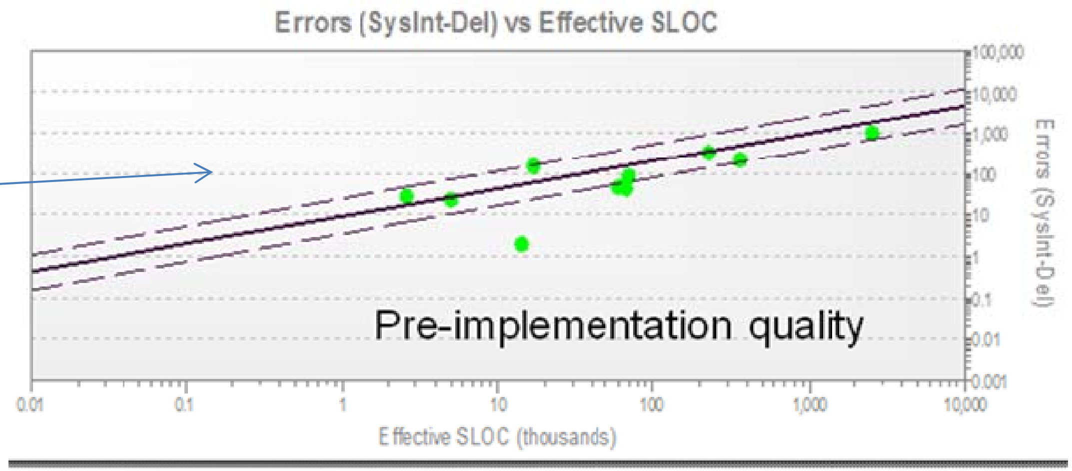
Best in class projects are green
Worst in class projects are red

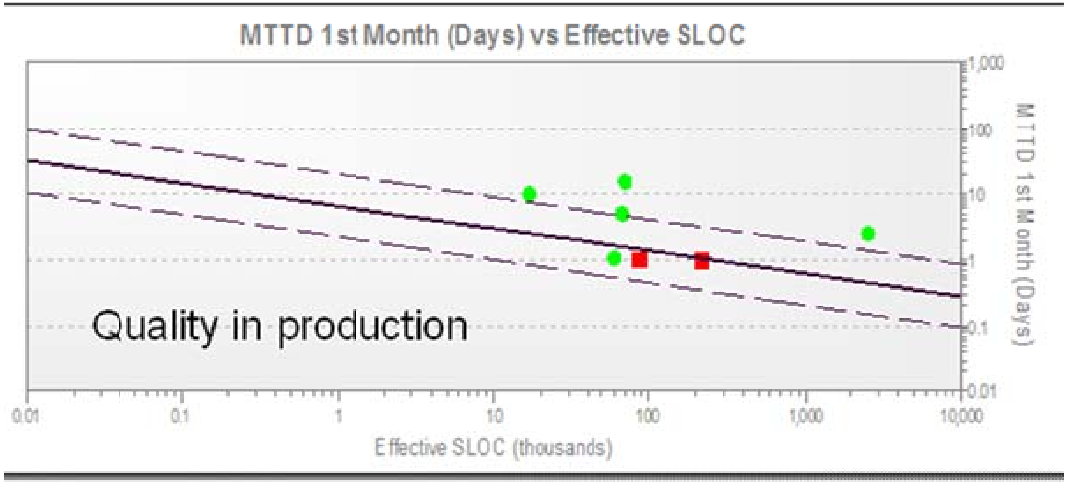
Best-in-Class Worst-in-Class Staffing



Best-in-Class Worst-in-Class Quality

Where are the Worst-in-Class projects?





Best-in-Class Worst-in-Class Industry

- 28% of projects in sample were for U.S. military
- 24% of projects were for aerospace industry
- 1 military project was Best-in-Class
- 80% of Worst-in-Class were for U.S. military

Characteristics

Best in Class

- Small teams
- More time & effort spent in feasibility & design
- Track & use defects during development
- Mostly non-military projects

Worst in Class

- Large teams
- Less time & effort in feasibility & design; more in post-implementation warranty
- Do not track & use defects during development
- Mostly military projects

Takeaways

- Time and effort spent upfront improve project performance
- Smaller teams are more effective
- Defect tracking is crucial to performance

Observations

- Small enhancements seem to be a "safe" development strategy (No Worst in Class projects)
- Best-in-Class projects use small teams; Worst-in-Class large ones
- Best-in-Class projects capture defects; Worst-in-Class do not
- High percentage of military projects in Worst-in-Class category cannot be completely attributed to the complexity or nature of the work

The Staffing Issue

Large teams have a minimal impact on schedule. Why?

Software Productivity Equation

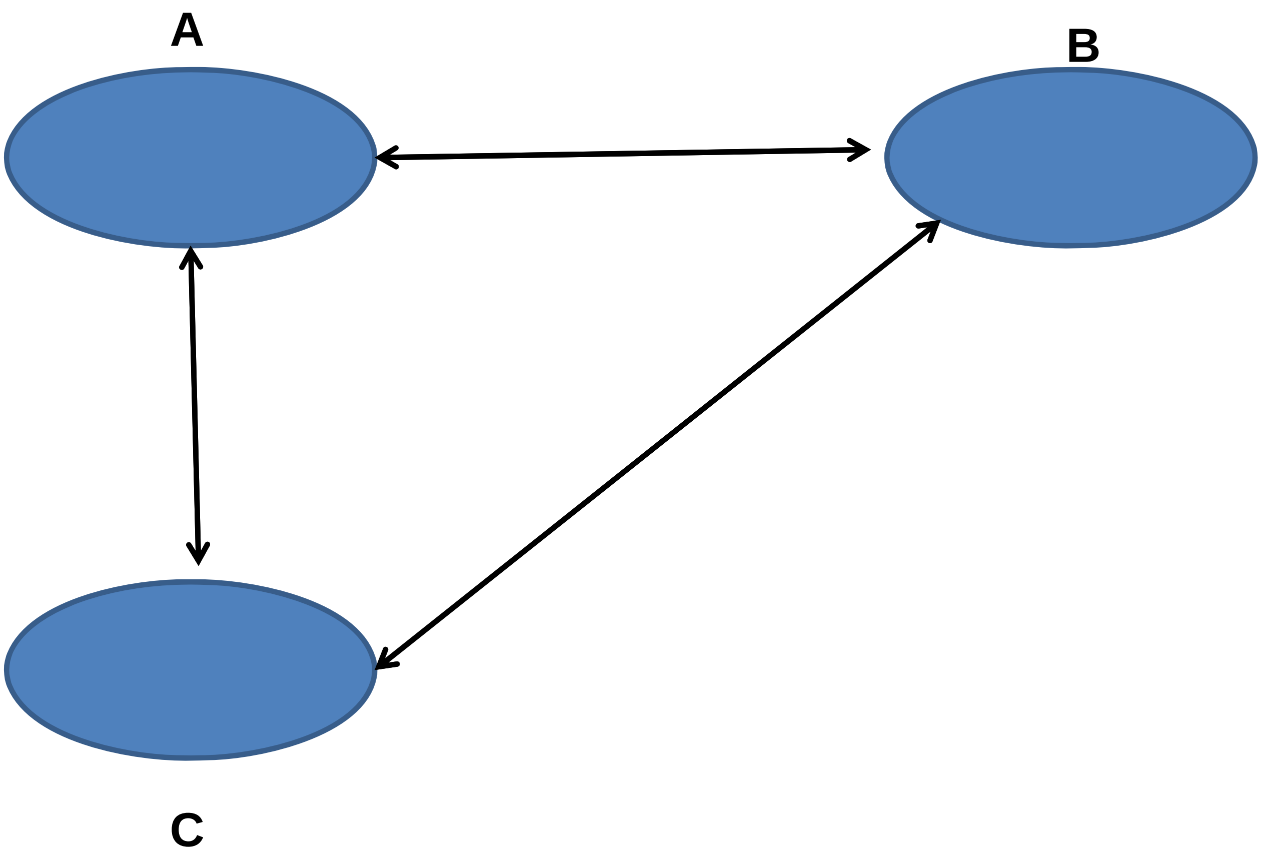
$$\text{Size} = \text{Effort}^{a} \times \text{Time}^{b} \times \text{Productivity}$$

where $a = \frac{1}{3}$ and $b = \frac{4}{3}$

Additional effort has a very modest impact on schedule or the amount of software that can be delivered

However, increasing staff will increase defects

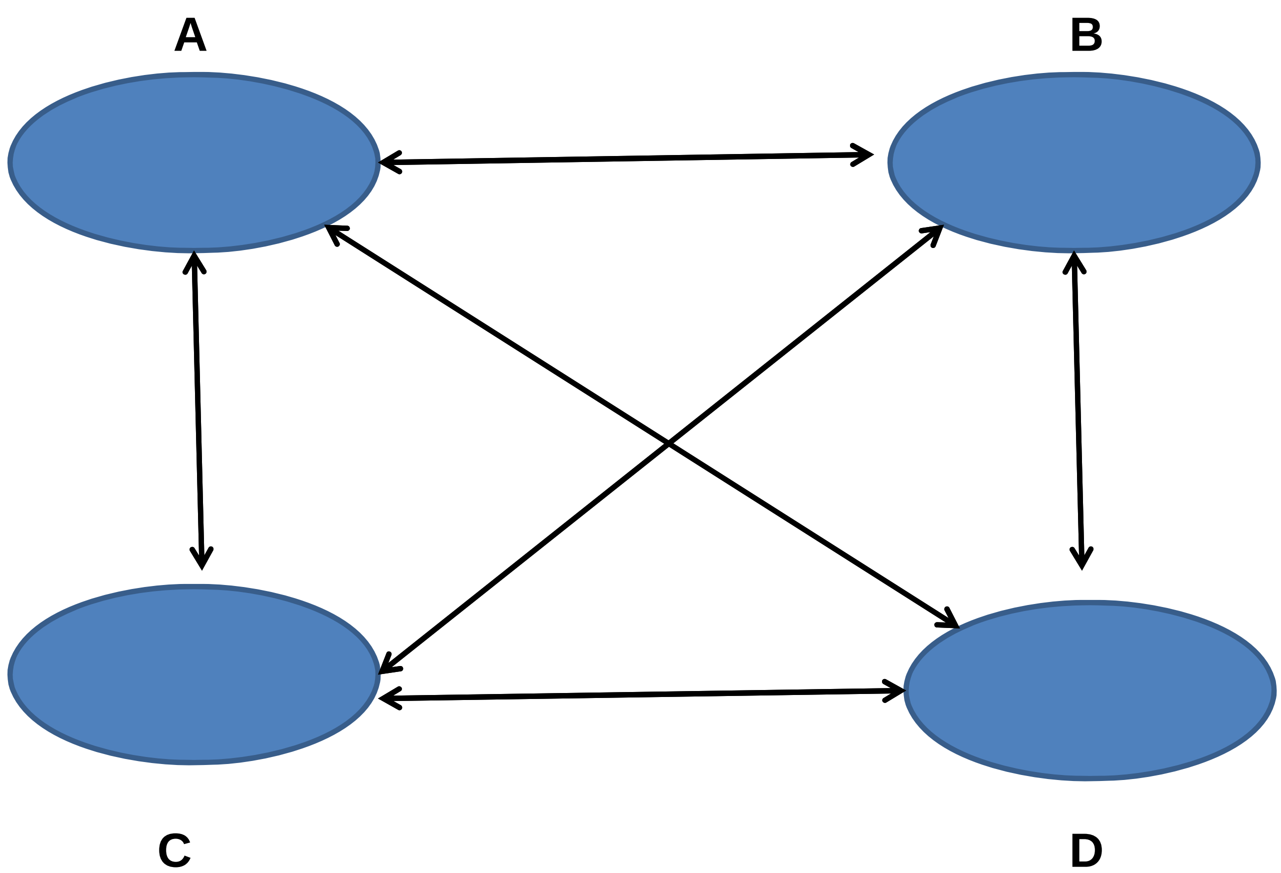
Staffing Example: a Three Person Team



Communication paths

- AB
- AC
- BC
- ABC

Staffing Example: a Four Person Team



Communication paths

- AB
- AC
- BC
- ABC
- BD
- CD
- AD
- ABD
- BCD
- CAD
- ABCD

Staffing in Summary

Increasing team size causes a non-linear increase in communication complexity

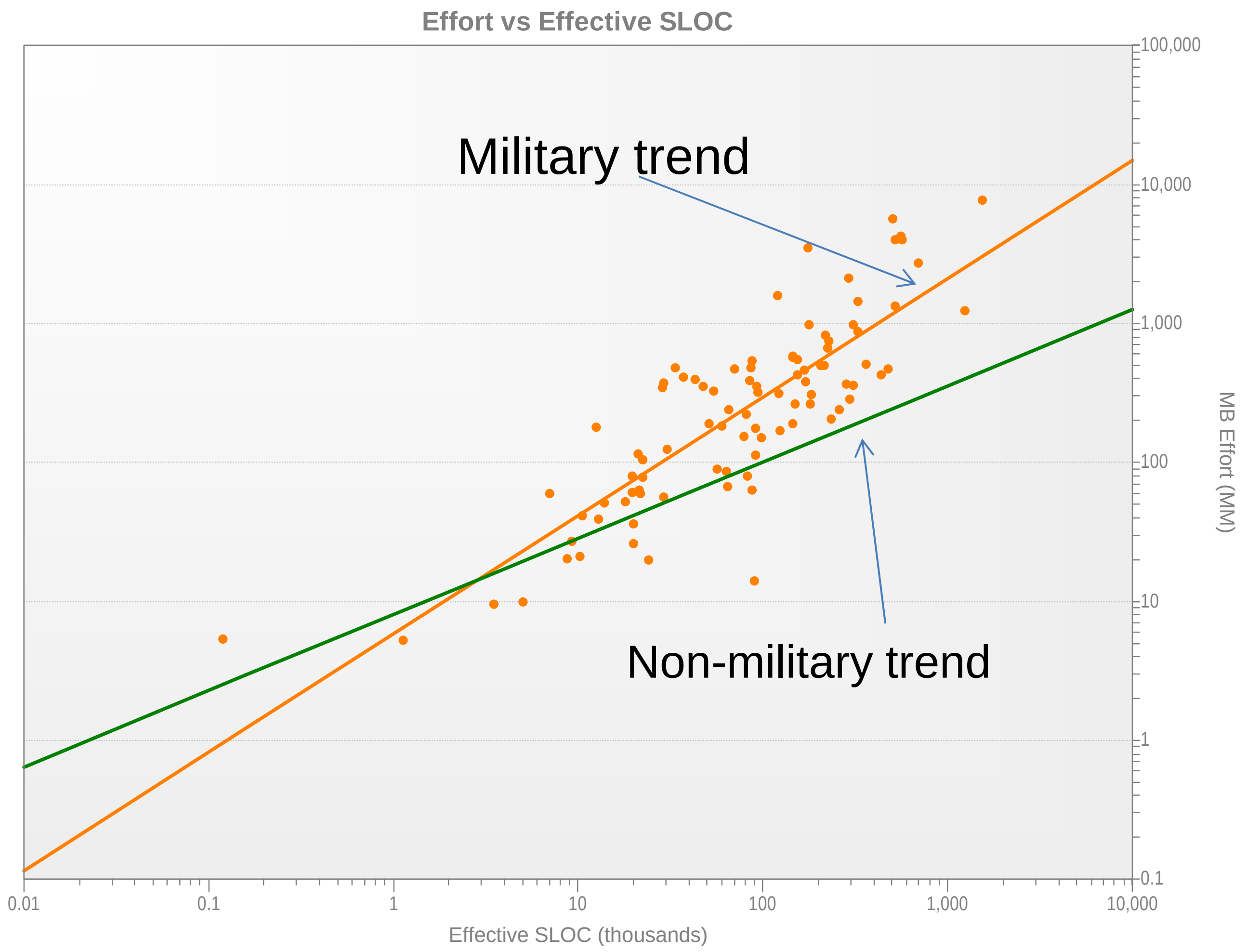
Which causes an increase in defects

Which causes an increase in re-work

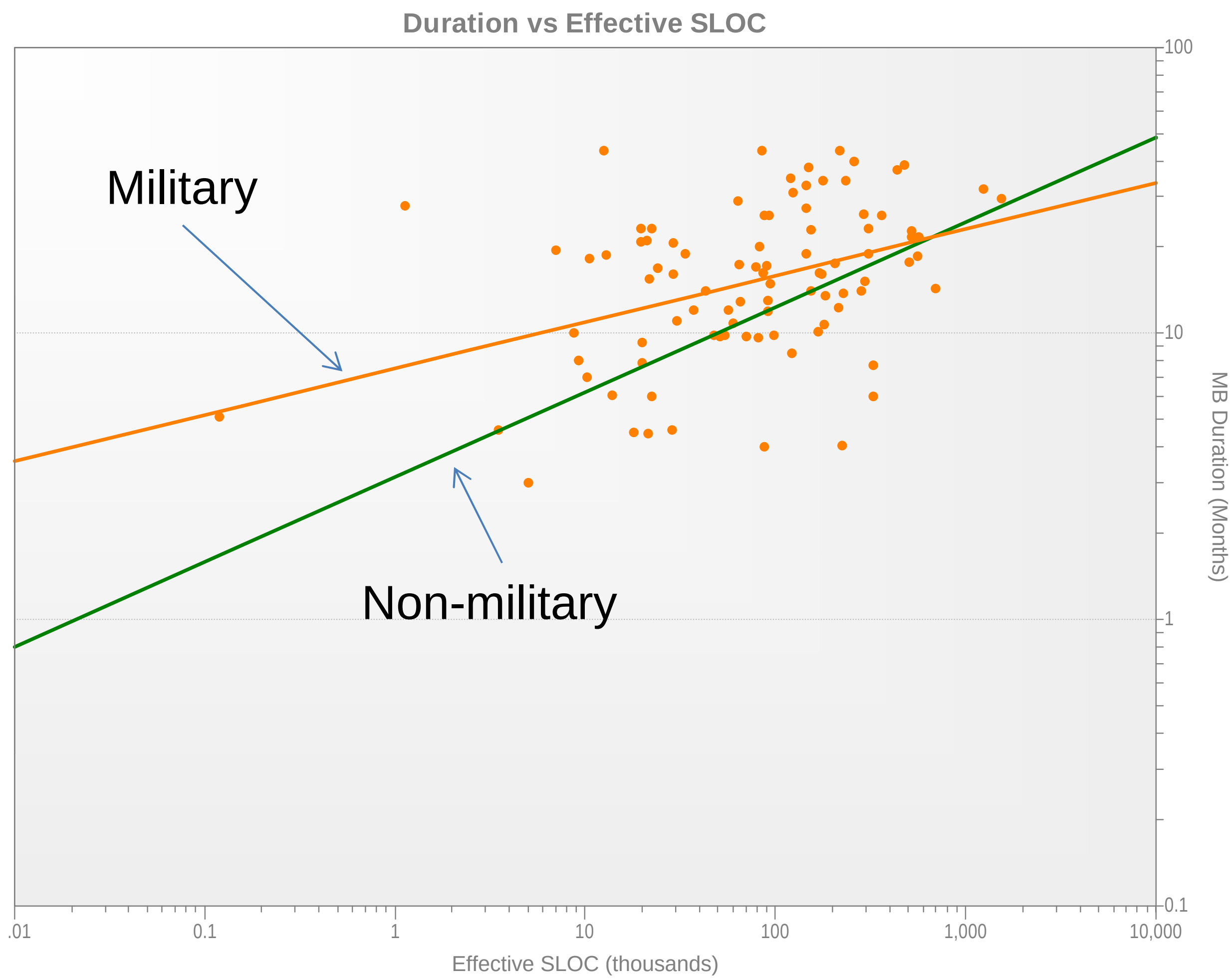
Which lowers productivity, lengthens schedule, and increases cost along with lowering team morale and increasing customer dissatisfaction

Recommendation: Use industry benchmarks to determine what is average staffing for your size of project as a starting point for staffing

Military vs. Non-Military Projects Effort



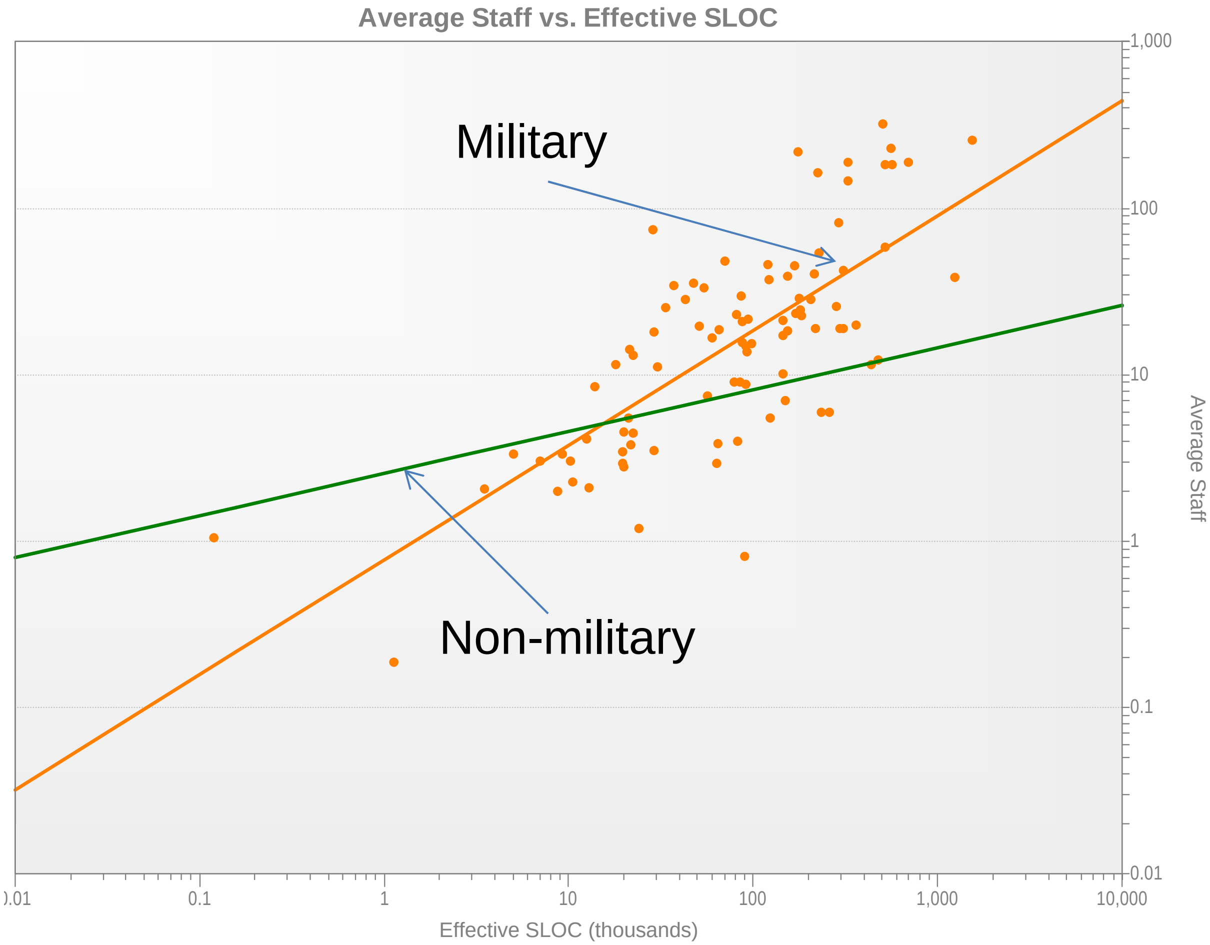
Military vs. Non-Military Projects Schedule



Military projects take longer to complete than non-military ones.

There seems to be little relationship between their schedules and the amount of functionality they deliver

Military vs. Non-Military Projects Average Staff



Military projects have significantly higher staffing levels than non-military ones

Dynamic Tension

Systems integrators want

- Win contracts
- Make a profit
- Achievable goals for cost, staff, schedule
- Stable requirements
- Satisfied customer
- Add-on or new work

Government wants

- Minimize cost and maximize benefit
- Competent, responsive systems integrators
- High quality systems delivered on time and within budget
- No surprises

Potential Gottcha's

- **Systems Integrator**: Winning contracts and making a profit can work against each other
- **Government**: Minimizing cost and maximizing benefits can work against each other
- **Both:** Contracts are awarded at a time when the requirements are at a level of detail that precludes precise planning
  - Government doesn't really know what it's asking for
  - Systems integrators don't really know what the expectations are nor what they are committing to

Possible Solutions

- **Know what is possible**
  - Government’s requirements for cost and effort should be demonstrably achievable based on past history
  - Systems integrators should know their capabilities based on their own historical performance and not commit to exceed them
  - Minimize staff
  - Maximize schedule flexibility
  - Break large projects into smaller releases that contain usable functionality (Idea borrowed from Agile)
  - Embrace uncertainty (See next slide)

Embracing Uncertainty
Project Scope

- Projects grow for 2 reasons:
  - Requests for additional functionality
  - Full implications of requirements become known
- Schedule, Staffing, Budget usually established based on higher level requirements
  - Big pictures seem simpler than they are. Full implications not known
  - Detailed requirements not fleshed out
  - Projects contain unknown unknowns (project "dark matter") that will impact size, schedule, and effort
- Metrics can help address this issue

Embracing Uncertainty
Project Scope

- Author's study (projects average 1 year duration)
  - 55% projects used more effort than planned, 26% less (average of 16% more)
  - 50% projects had longer schedules, 16% shorter (average 8% longer)
  - 90% projects were larger than planned, 10% smaller (average 15% larger)
- Rule of thumb
  - 1 year project: Increase scope (size) 18%
  - 2 year project: Increase scope 39%
  - 3 year project: Increase scope 64%

A Way Forward

- Since there is uncertainty around project scope and productivity, actual progress against plan should be monitored closely using empirical measures
  - Ideally, this should be a joint activity or be done by a third party
  - Plans should account for system growth that will occur
  - Plans should be updated when thresholds for deviation are exceeded
  - Measures to be reported and the frequency of reporting should be agreed upon before the project begins

Measures Needed to Effectively Monitor Project Performance

- A project plan that contains
  - Estimate of the functionality to be developed (size)
  - Monthly staffing plan
  - Major milestones and their dates
  - Overall schedule
- Monthly reporting that includes
  - Actual functionality developed (SLOC, Function Points, RICEF objects, etc.)
  - Actual FTE staffing
  - Dates when milestones actually occur
  - Defects discovered during reporting period
  - Defects resolved during reporting period

Questions?