What Happens and How: Analyzing the Results of 13 Acquisition Program Assessments

William E. Novak

Dr. Forrest J. Shull

Software Solutions Conference 2015

November 16–18, 2015

Software Engineering Institute | Carnegie Mellon University

Agenda

Introduction

Recurring Themes across Programs

Misaligned Incentives

Acquisition Archetypes

Breaking the Pattern

Solving Social Dilemmas

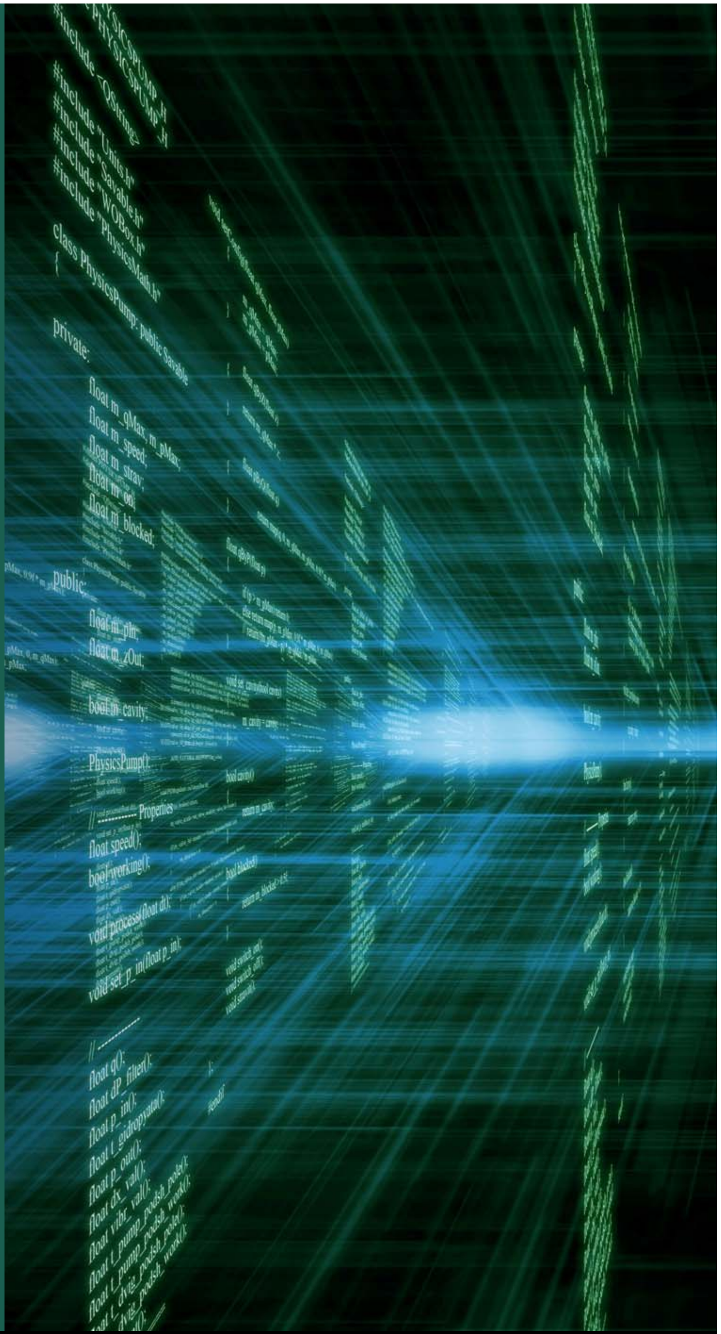
Summary and Conclusions

Analyzing the Results of 13
Acquisition Program Assessments

Introduction

"Large-system programming has over the past decade been… a tar pit, and many great and powerful beasts have thrashed violently in it. Most have emerged with running systems—few have met goals, schedules, and budgets. Large and small, massive or wiry, team after team has become entangled in the tar. No one thing seems to cause the difficulty—any particular paw can be pulled away. But the accumulation of simultaneous and interacting factors brings slower and slower motion. Everyone seems to have been surprised by the stickiness of the problem, and it is hard to discern the nature of it. But we must try to understand it if we are to solve it."

—Frederick Brooks, The Mythical Man-Month



Introduction

SEI Independent Technical Assessments (ITAs)

SEI conducts Independent Technical Assessments (ITAs) on large software-reliant acquisition programs

- *ITAs are objective program reviews of people, programmatics, processes, technical aspects, and the environment*

ITA teams conduct interviews & review documents on program status/history

- *Identify likely causes of schedule, cost, or performance issues*
- *Recommend improvement or recovery actions*

SEI brings to the assessments:

- *Software, systems engineering and program management expertise*
- *Independent and objective third-party perspective*
- *Experience in conducting over 100 ITAs and Red Teams*

Introduction

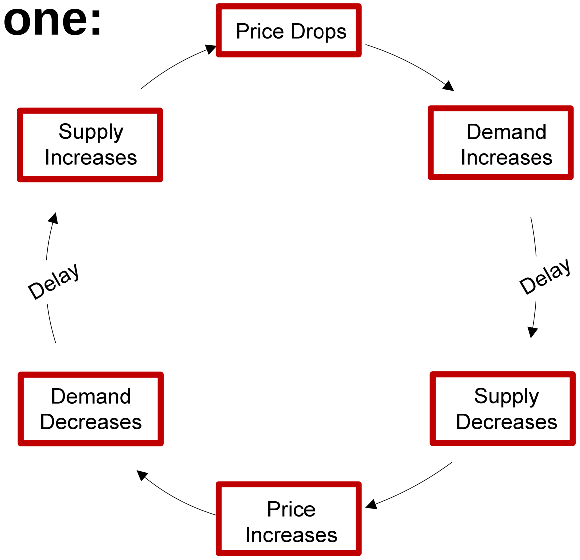
Can Systems Trap Us into Behaviors?

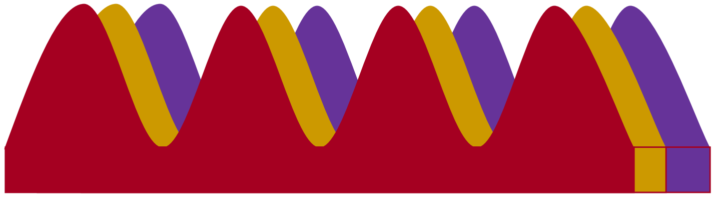
Inside a complex, dynamic system, people’s actions can be at the mercy of that system’s dynamics. Such patterns occur in real estate cycles:

As price drops…

→ demand increases	(*get a good deal*)
→ *…and after a delay…*	(*takes time to buy*)
→ supply decreases	(*not many houses left*)
→ price *increases*	(*supply and demand*)
→ demand *decreases*	(*too expensive now*)
→ *…and after a delay…*	(*more people must sell*)
→ supply *increases*	(*plenty of houses*)
→ and price drops…	(*supply and demand*)

Since this is a *loop*, let’s draw it as one:

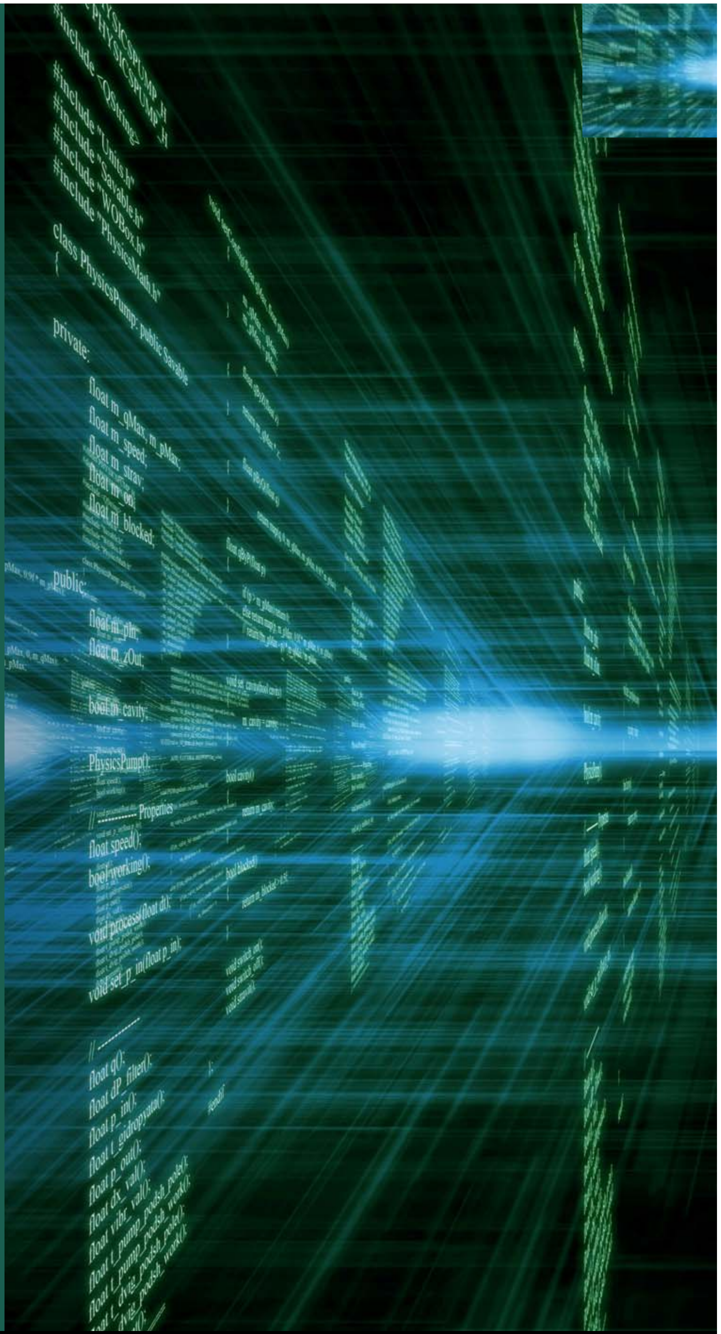




Analyzing the Results of 13 Acquisition Program Assessments

Recurring Themes across Programs

Results from work with:
Rhonda Brown
Michael Konrad



Software Engineering Institute | Carnegie Mellon University

Findings / Recurring Themes

Goals of the Analysis

Leverage SEI participation in a broad cross-section of DOD programs to reflect on recurring challenges with which DOD programs are dealing

- Examine unclassified SEI ITAs performed in the past five years, for which out-briefings or reports of findings are available for review
- Determine the major software-related challenges DoD programs are facing
- Use analysis results as input for prioritizing data-driven research in SEI

Findings / Recurring Themes

Analysis Methodology

ITA outbriefs and supporting documents were analyzed and text segments representing challenges ("findings") were flagged.

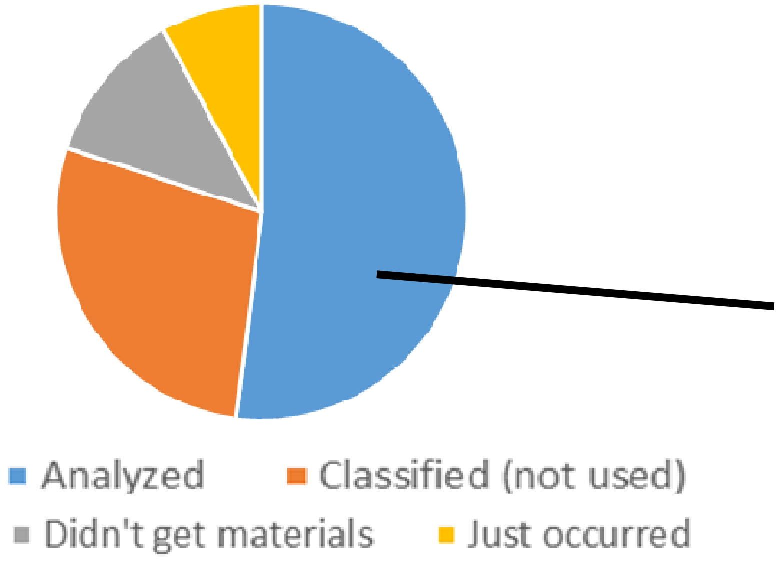
- Each finding was mapped to a taxonomy, representing a hierarchy of issue types. New categories were created, if needed, as new types of issues were found.
- After an initial pass, several ITAs were reviewed independently by more than one analyst and the results were discussed until consensus on classification was gained. This led to several changes to the taxonomy and re-definition of several categories.
- The "findings taxonomy" that emerged was used to identify and understand trends across ITAs.
- All marking and categorization was done in the Nvivo tool.

The ordering of the top10 software challenges was determined based on:

1. The number of **programs** affected by the issue (indicator of *prevalence*)
2. The number of **findings** mapped to each issue - only if a tie-breaker was needed (weak indicator of *intensity*)

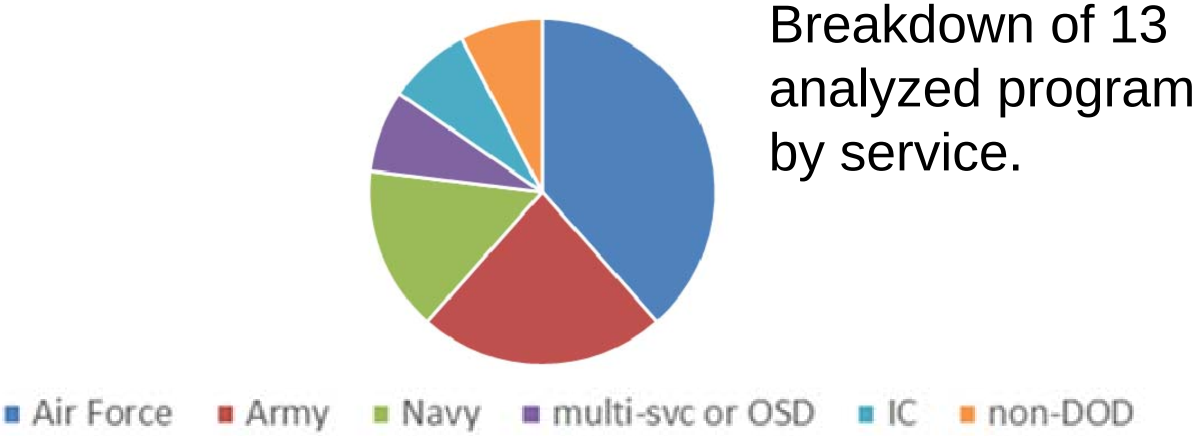
Findings / Recurring Themes

Data Set: ITAs Conducted 2009-2013



25 ITAs conducted in that time period

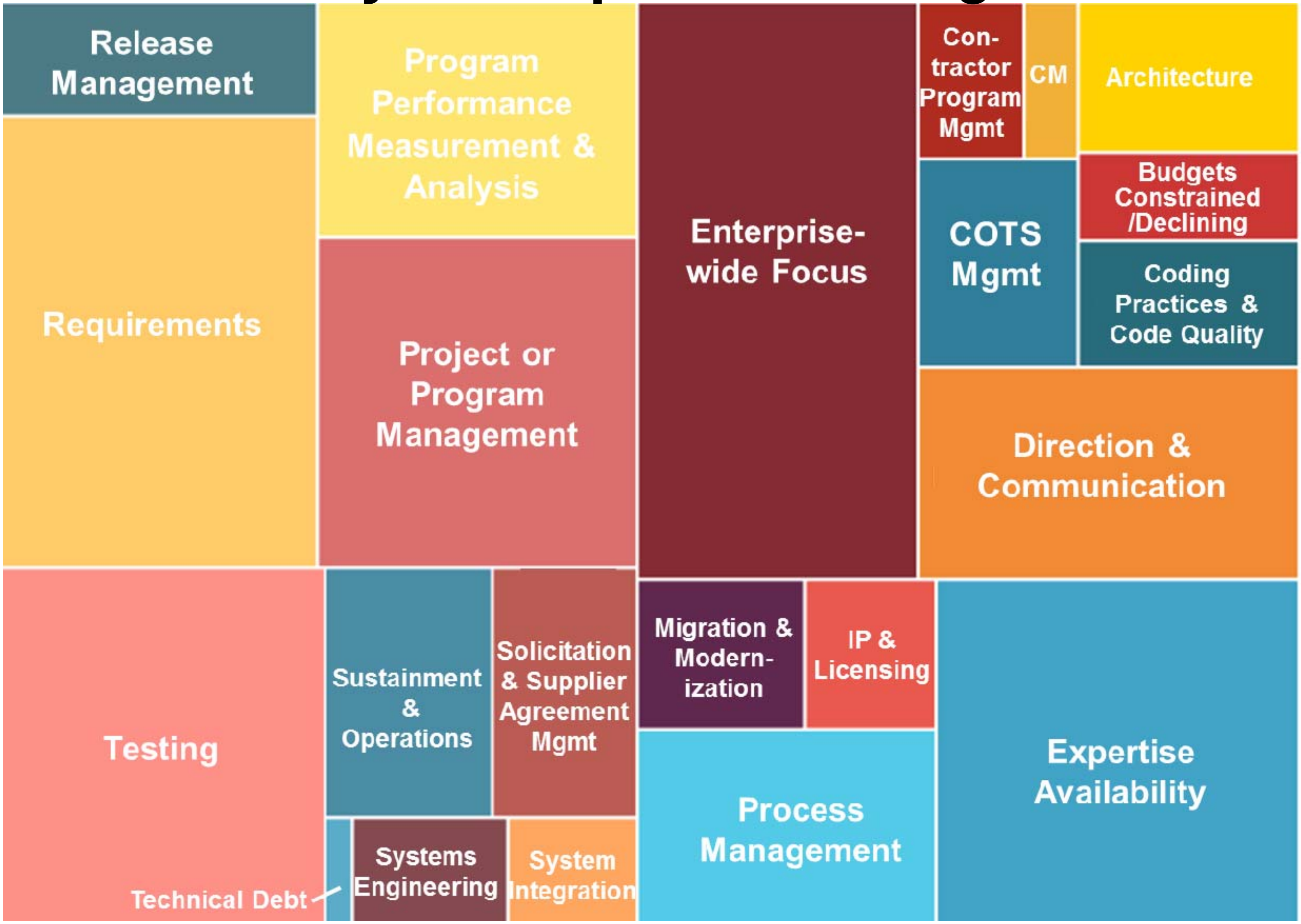
13 ITAs / Programs analyzed



Breakdown of 13 analyzed programs by service.

Findings / Recurring Themes

Taxonomy: 22 Top-Level Categories



449 findings in total

(Size of box represents number of findings)

Taxonomy:

- 22 top-level categories
- 114 categories total

Findings / Recurring Themes

Top ITA Issues by Program

		Program												
Issues	**# Pgms**	1	2	3	4	5	6	7	8	9	10	11	12	13
1.Alignment / program focus	10		x	x	x	x	x	x	x	x			x	x
2.Direction and communication	8		x	x	x		x	x	x	x				x
3.Sufficient measures of progress	6	x	x	x				x		x				x
4.Documented software architecture	6		x	x	x	x			x	x				
5.Release management	6		x		x					x	x	x		x
6.Staff planning and training	6		x		x	x		x		x				x
7.Organizational structure	5		x					x	x	x	x			
8.Requirements change drivers and traceability	5			x			x		x		x			x
9.Program oversight	5		x		x			x		x				x
10.Processes defined and integrated	5				x	x		x	x	x				

Findings / Recurring Themes

#1: Alignment

10 out of 13 ITAs

The complexity of today’s systems means that their development is conducted by multiple organizations with different areas of expertise and different responsibilities.

Alignment describes the degree to which technical activities across organizational structures cooperatively support the program goals and vision.

Findings / Recurring Themes

#1: Alignment

Issues encountered:

- Expertise not available or utilized in relevant lifecycle stages or components
- Finger-pointing rather than shared responsibility
- Competition among program components for resources
- Adding new capabilities prioritized inappropriately over continuity of operations
- Institutional barriers that separate the user community from developers
- No effective conflict resolution mechanism

Results most often cited:

- Substantial disparities in evaluating delivered quality and program status
- Dissatisfied users
- Relevant perspectives left out of early lifecycle phases, with potential problems for later phases
- Fiscal discipline hard to enforce
- Long-range planning hampered
- A single stakeholder perceived as wielding too much influence over joint solutions

Findings / Recurring Themes

#2: Direction and Communication

8 out of 13 ITAs

Effective program management involves the use of regular, efficient communication to coordinate the program, especially (but not exclusively) across organizational boundaries.

Clear and regular lines of communication help ensure that personnel are fully aware of actions and events at other locations, or within other parts of the program.

A shared understanding of priorities and technical progress among stakeholders, both internal and external to the program, is an expected output.

Findings / Recurring Themes

#2: Direction and Communication

Issues encountered:

- ITAs found poor communication of:
  - Changing priorities of requirements or goals
  - Requirements changing due to status of the program (e.g. from “developmental” to critical system, from system under development to one in sustainment)
  - Division of roles and responsibilities among organizations
- Exacerbated by contracting constraints (real or perceived)

Results most often cited:

- Lack of true understanding of program status
- Attention and effort focused on wrong things
- Lack of true program-wide focus
- Adverse effect on routine operations
- Rework and late detection of errors

Findings / Recurring Themes

#3: Measures of progress

6 out of 13 ITAs

Given the long development duration and complexity of these systems, the government acquisition office needs to be skilled in monitoring progress. Metrics definition, collection, and analysis are important tools in this regard.

Successful measurement provides an insightful and accurate picture of status across many years and different lifecycle phases, sufficient to support effective decision-making.

Findings / Recurring Themes

#3: Measures of progress

Issues encountered:

- No progress metrics on some programs:
  - Project metrics are unavailable to monitor contractor development status/quality
  - No hard data to verify perceptions of effectiveness and efficiency
  - No documented metrics collection and analysis effort
- Inadequate metrics or use of metrics on others:
  - Inconsistent software metrics collected
  - Limited progress measurement for essential acquisition insight of software
  - More data is collected than is used for oversight
  - Not clear the right data is requested, collected, reviewed, or understood
  - Unclear purpose for data collection

Results most often cited:

- Inaccurate estimates
- No program-level insight
- Inability to monitor contractors effectively
- Surprises in cost, schedule, and technical performance

Findings / Recurring Themes

#8: Reqts. Changes & Traceability

5 out of 13 ITAs

Requirements changes have long been recognized as a source of program challenges, especially when such changes occur late in the lifecycle. Avoidable problems (and their associated rework) can stem from deficiencies in both functional and non-functional requirements. The process used for requirements analysis can have a substantial impact on the completeness of the requirements produced.

Modern programs should find opportunities to proactively consider user requirements, but are also affected by legislative and policy changes that have implications for program capabilities or management processes.

Omission of these requirements creates unrealistic assumptions that greatly impact cost and schedule, especially when release schedules are aggressive, involving concurrent, independent activities.

Findings / Recurring Themes

#8: Reqts. Changes & Traceability

Issues encountered:

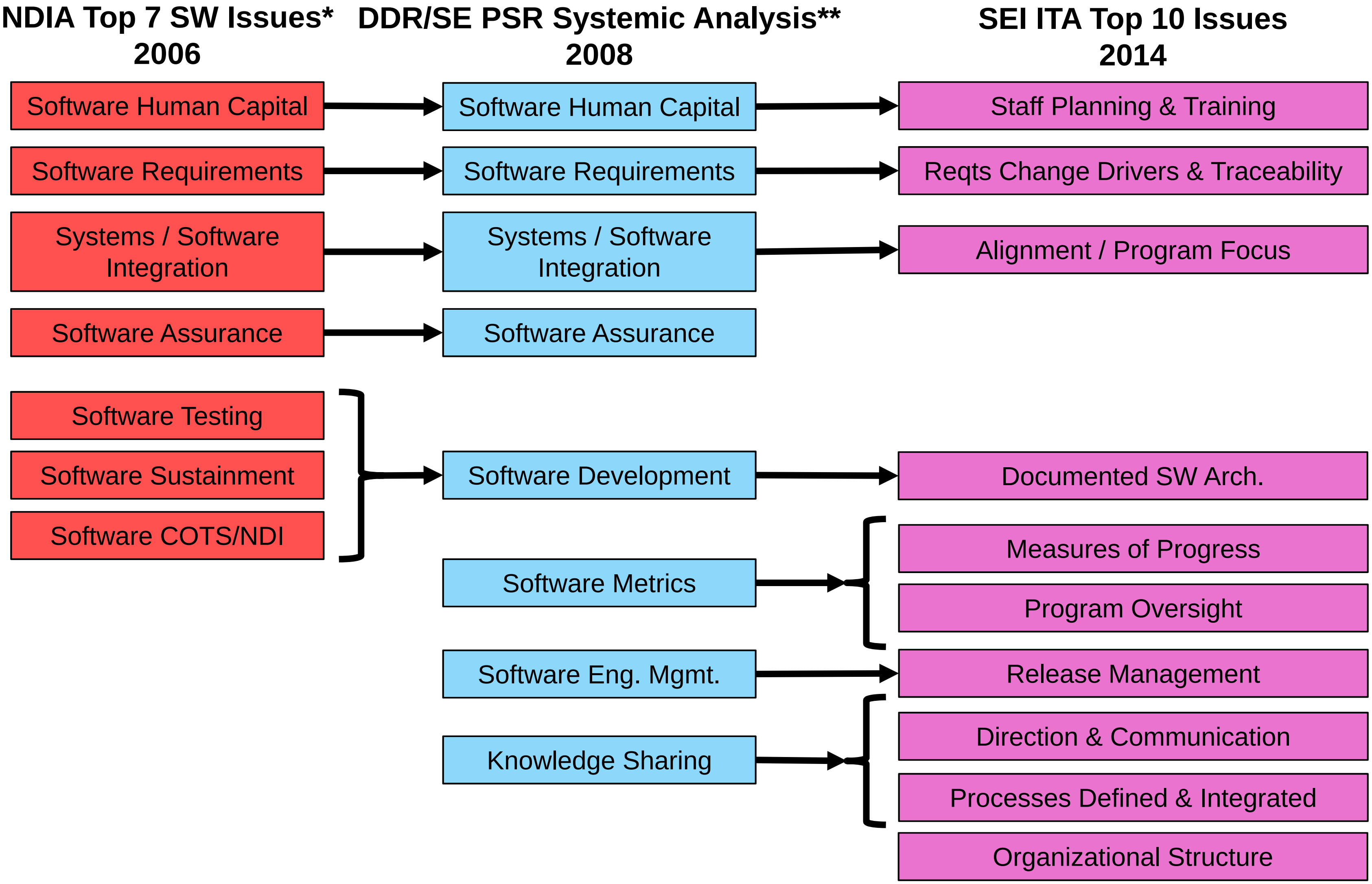
- Un-prioritized and ambiguous requirements
- Numerous undocumented requirements to evaluate
- Interface requirements across systems (among services) not identified
- Lack of reliability (and other non-functional) requirements
- Many systems requirements (stress, capacity, loading) not identified
- Unexpected scope changes late in development

Results most often cited:

- Late rework due to lack of early stakeholder input
- Contractual acceptance of deficient requirements deliverables creates the [false] expectation that the requirements process was complete and all requirements were completely captured.

Findings / Recurring Themes

Mapping to Other Software Issues Lists



NDIA Top 7 SW Issues* 2006

- Software Human Capital
- Software Requirements
- Systems / Software Integration
- Software Assurance
- Software Testing
- Software Sustainment
- Software COTS/NDI

DDR/SE PSR Systemic Analysis** 2008

- Software Human Capital
- Software Requirements
- Systems / Software Integration
- Software Assurance
- Software Development
- Software Metrics
- Software Eng. Mgmt.
- Knowledge Sharing

SEI ITA Top 10 Issues 2014

- Staff Planning & Training
- Reqts Change Drivers & Traceability
- Alignment / Program Focus
- Documented SW Arch.
- Measures of Progress
- Program Oversight
- Release Management
- Direction & Communication
- Processes Defined & Integrated
- Organizational Structure

* http://www.ndia.org/divisions/divisions/systemsengineering/documents/studies/ndia_top_sw_issues_2006_report_v5_final.pdf

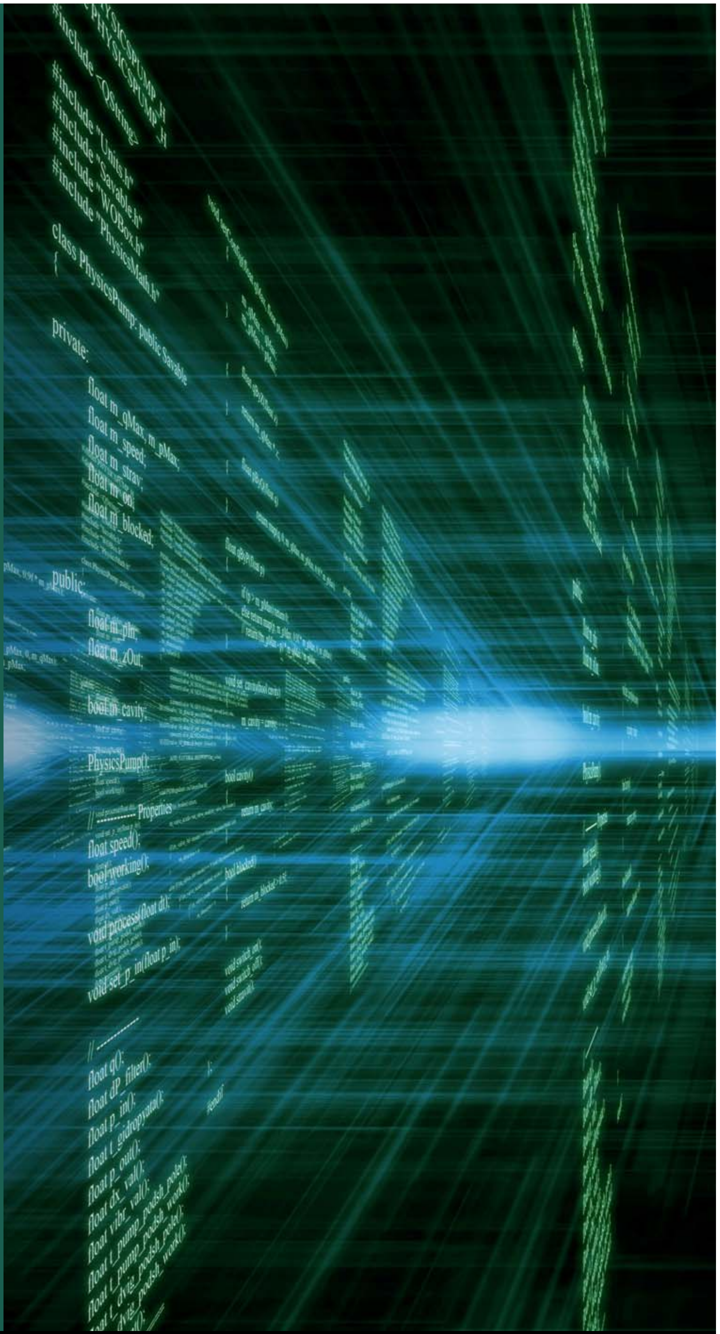
** http://www.dtic.mil/ndia/2009systemengr/8977ThursdayTrack8Lucero.pdf

Analyzing the Results of 13
Acquisition Program Assessments

Misaligned Incentives

“Incentives are misaligned—PMs and contractors are not necessarily rewarded for decisions that lead to lower life cycle costs or provide a better balance between cost and performance”

—Defense Acquisition Performance Assessment,
GEN Ronald Kadish (Ret.)



Software Engineering Institute | Carnegie Mellon University

Social Dilemmas

Misaligned Incentives

Structural reasons like feedback and delays aren’t the only causes for acquisition failure—*incentives* play a key role as well.

Misaligned incentives occur when:

- *Individual* goals conflict with *group* goals
- *Short-term* goals conflict with *longer-term* goals

The result is that:

- Some *group* goals only succeed at the expense of *individual* goals
- Some *longer-term* goals can only succeed at the expense of *short-term* goals

Some acquisition programs are *prevented* from succeeding for structural and incentive reasons—*not* poor work or lack of effort.

Take-away

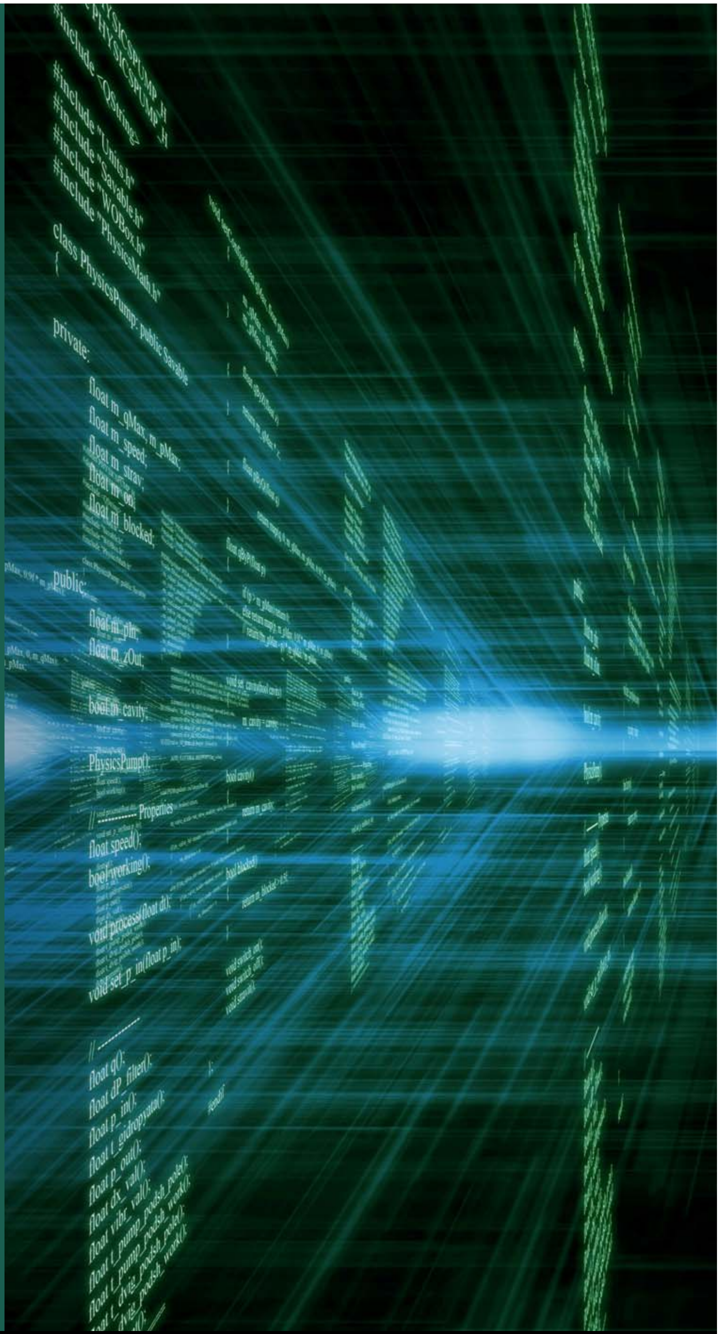
Misaligned incentives can force people to make impossible choices.

Analyzing the Results of 13 Acquisition Program Assessments

Acquisition Archetypes

We’re “…focused on the little, tiny swells and waves on the surface of the ocean. But in fact, most of the big things affecting the ocean are these currents underneath. They're what's moving the water.”

—*John Sides, GWU*



Software Engineering Institute | Carnegie Mellon University

What are Acquisition Archetypes?

Acquisition archetypes are modeled on the systems archetypes

Acquisition archetypes are patterns of behavior seen time and again on actual programs that are counter-productive and undermine progress

Acquisition archetypes depict the underlying structures of the behaviors that occur throughout acquisition organizations

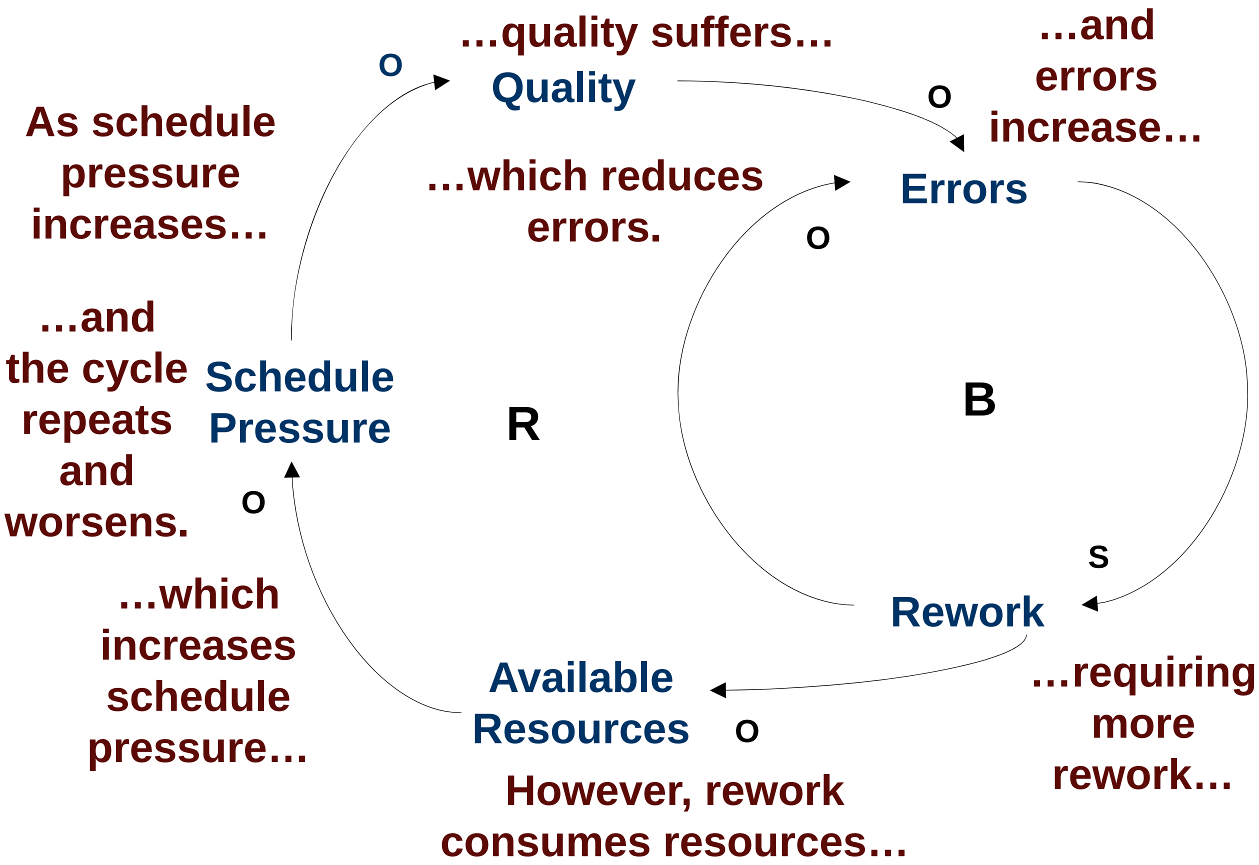
- Each causal loop diagram tells a familiar, recurring story
- Each describes the structure that causes the dynamic

Acquisition Archetypes are used to:

- Identify failure patterns as they develop (*recognition*)
- Single out root causes (*diagnosis*)
- Engage in “big picture” thinking (*avoid oversimplification*)
- Promote shared understanding of problems (*build consensus*)
- Find interventions to break out of ongoing dynamics (*recovery*)
- Avoid future counter-productive behaviors (*prevention*)

Acquisition Archetypes

“Happy Path Testing” (*i.e.*, *Sacrificing Quality*)



As schedule pressure increases…

…quality suffers…

…and errors increase…

…which reduces errors.

…requiring more rework…

However, rework consumes resources…

…which increases schedule pressure…

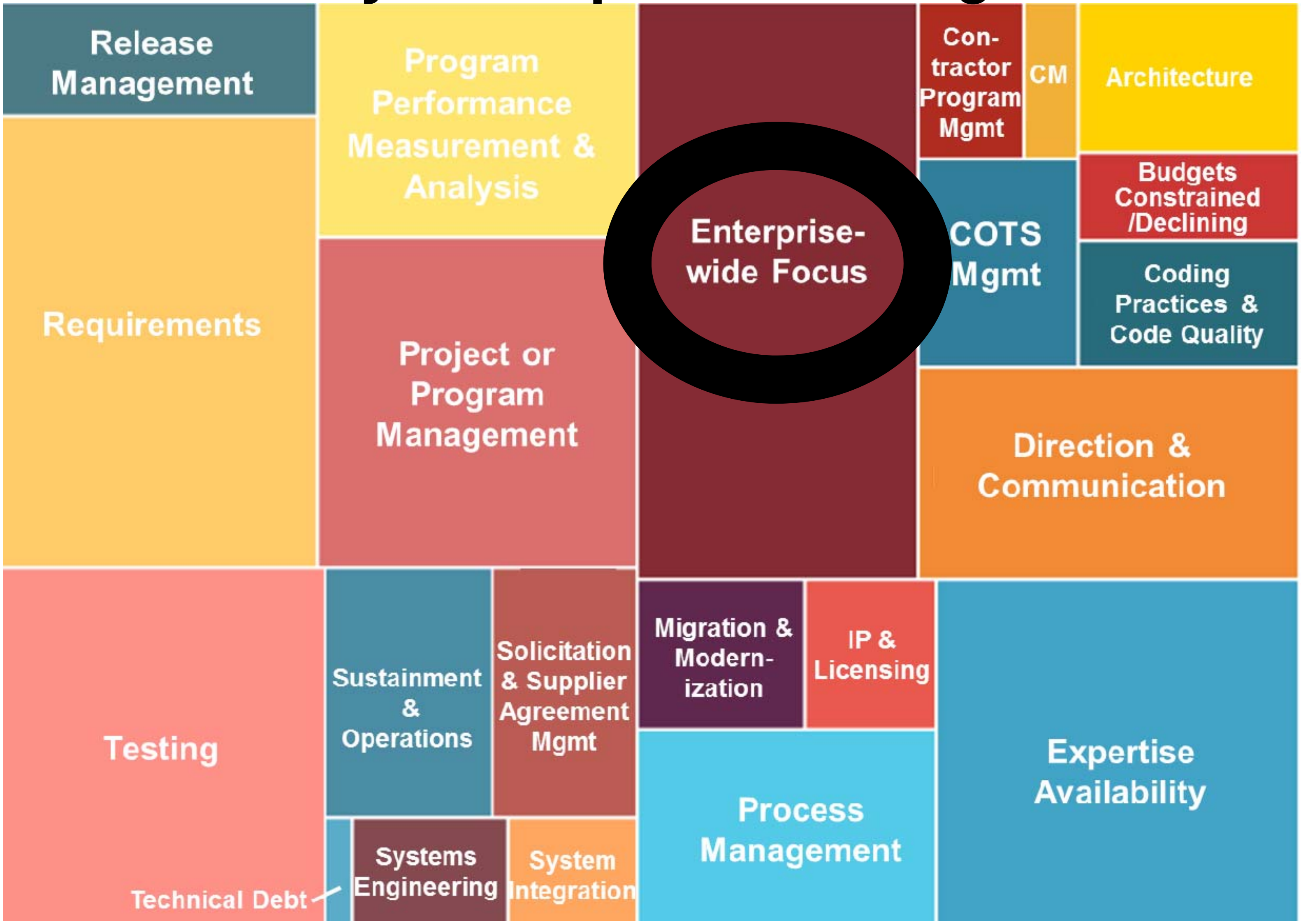
…and the cycle repeats and worsens.

As schedule pressure increases, processes are shortcut, quality suffers, and errors increase—requiring more re-work. However, re-work consumes resources, which increases schedule pressure, and the cycle repeats and worsens.

based on the “Fixes that Fail” systems archetype

Acquisition Dynamics

Taxonomy: 22 Top-Level Categories



449 findings in total

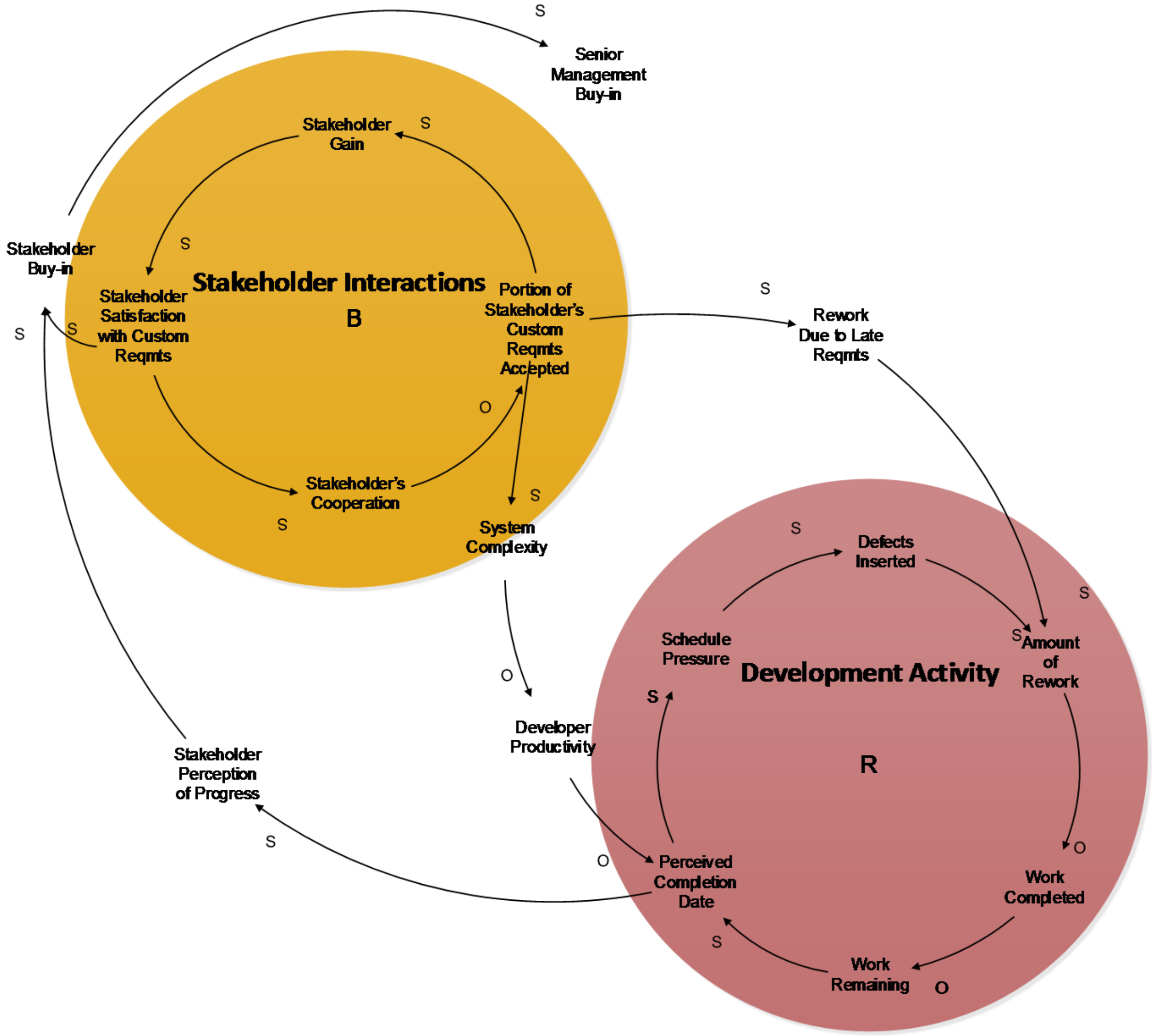
(Size of box represents number of findings)

Taxonomy:

- 22 top-level categories
- 114 categories total

Enterprise-Wide Focus

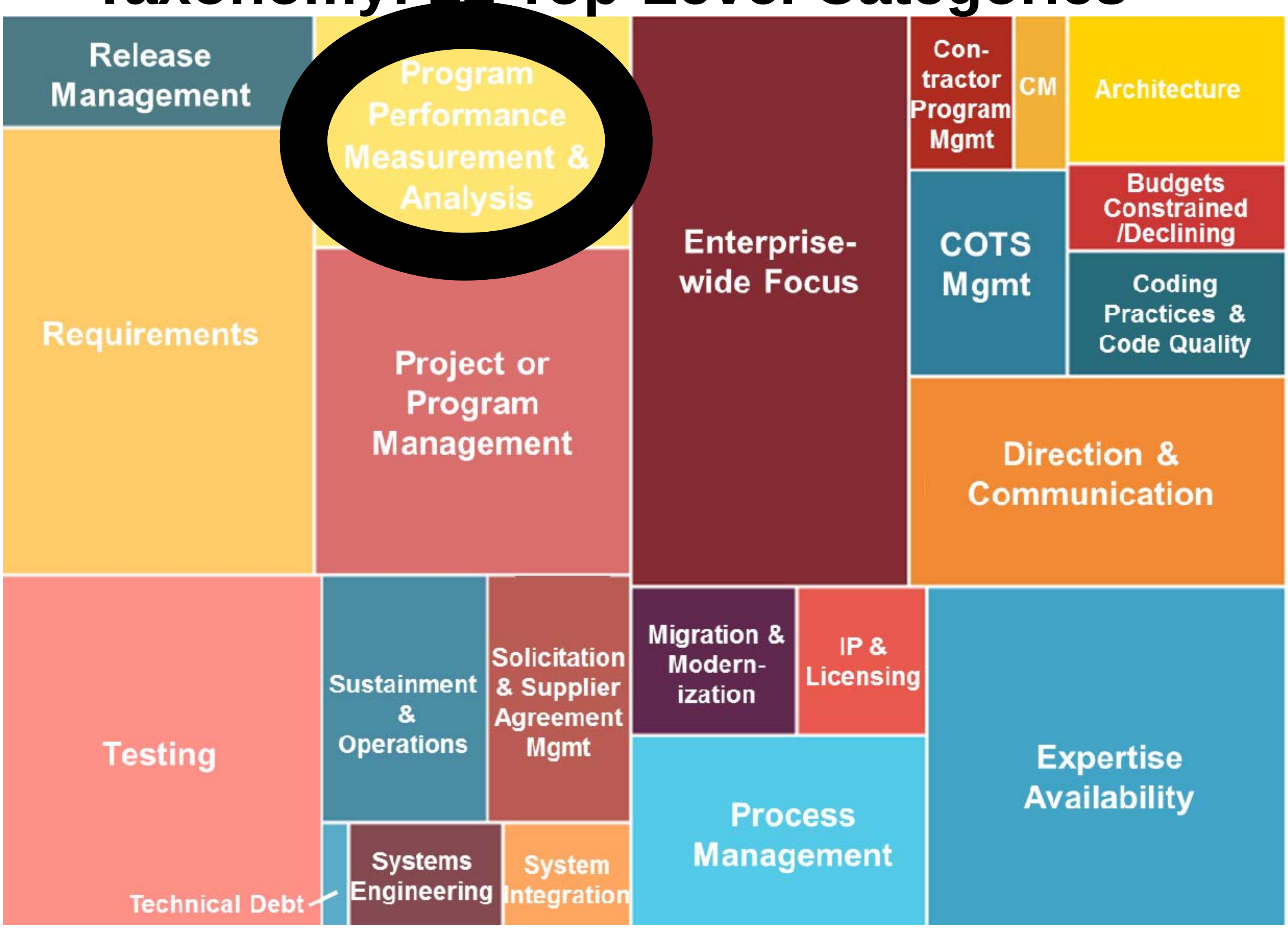
“Common Infrastructure”



A common infrastructure software development has multiple stakeholders planning to use it. As work proceeds, each stakeholder demands a new feature be added to meet their unique requirement. The PM is reluctant to add them, but agrees to help keep stakeholders on board. The new changes and coding drive up total cost, schedule, complexity, and risk. As schedule slips, one stakeholder realizes they will now miss an upcoming deadline—and so leaves the infrastructure effort. With one stakeholder (and their funding) gone, the cost for the others increases. The growing delays now impact other stakeholders as they approach their deadlines, and another stakeholder leaves as well. Costs rise further, along with delays, and the remaining stakeholders follow suit, moving to use custom solutions—and the common infrastructure effort unravels and collapses.

Acquisition Dynamics

Taxonomy: 22 Top-Level Categories



449 findings in total

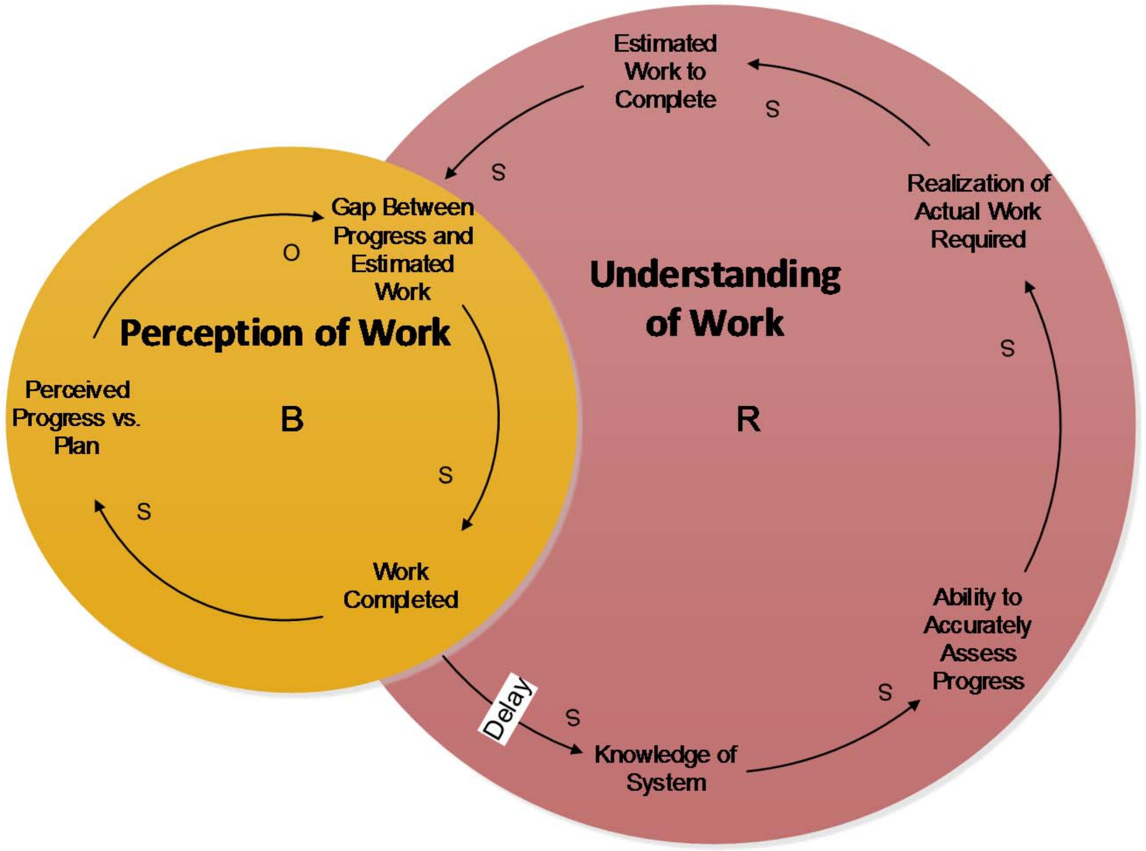
(Size of box represents number of findings)

Taxonomy:

- 22 top-level categories
- 114 categories total

Program Performance Measurement & Analysis

“The 90% Syndrome”

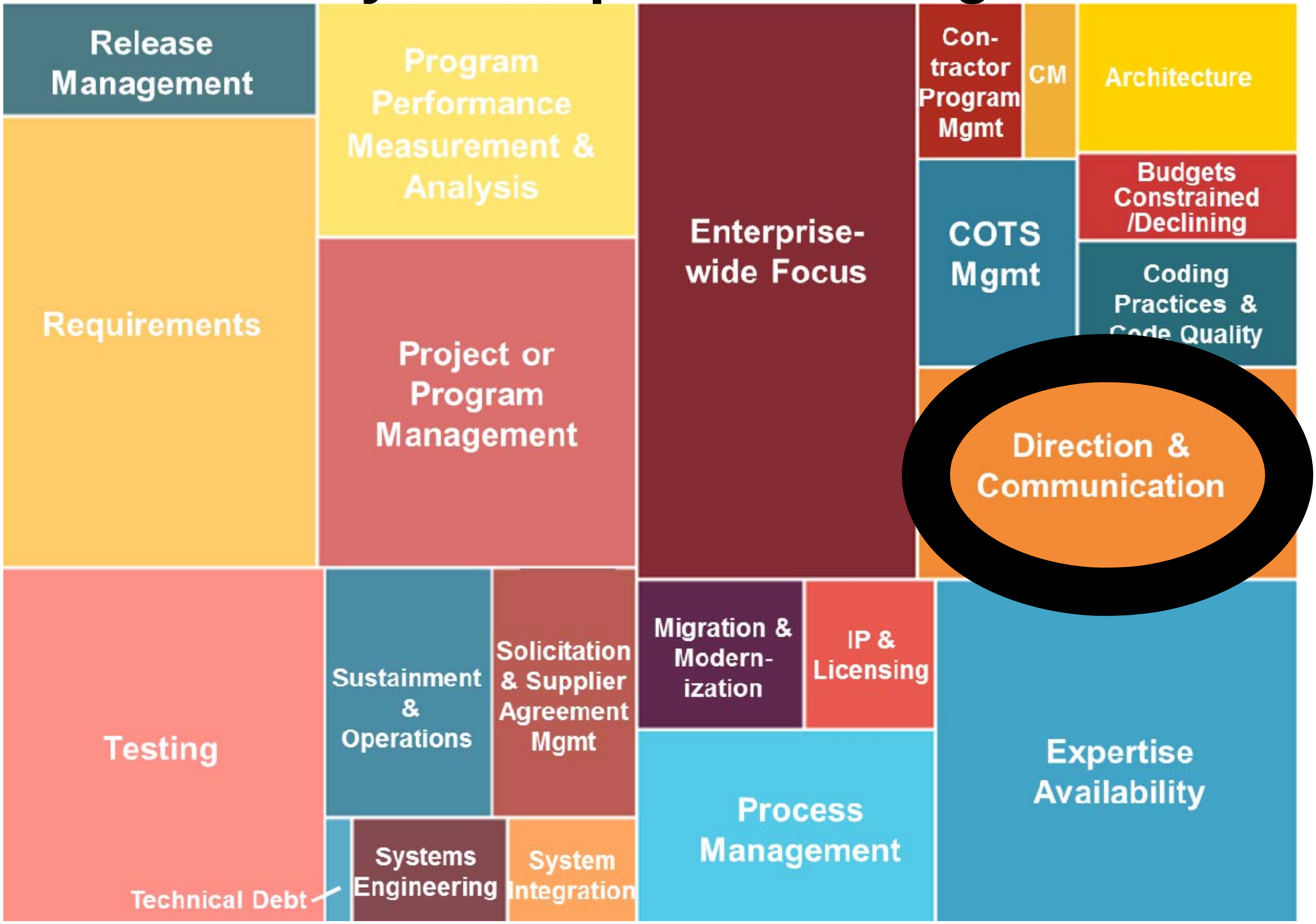

Estimated Work to Complete
S
S
Realization of Actual Work Required
Gap Between Progress and Estimated Work
O
Understanding of Work
Perception of Work
S
Perceived Progress vs. Plan
B
R
S
S
Work Completed
Ability to Accurately Assess Progress
Delay
S
S
Knowledge of System

A software project initially has only crude measurements of its progress, measuring resources spent instead of actual progress. As work progresses, knowledge of the system’s degree of completeness vs. the amount of work remaining improves, and the growing gap between planned and actual progress becomes apparent. As the realization of the actual work remaining becomes clearer, progress appears to stall at the 80%-90% done threshold, despite the amount of real work that is being done.

Concept from “Software Project Dynamics” by Tarek Abdel-Hamid & Stuart Madnick

Acquisition Dynamics

Taxonomy: 22 Top-Level Categories



449 findings in total

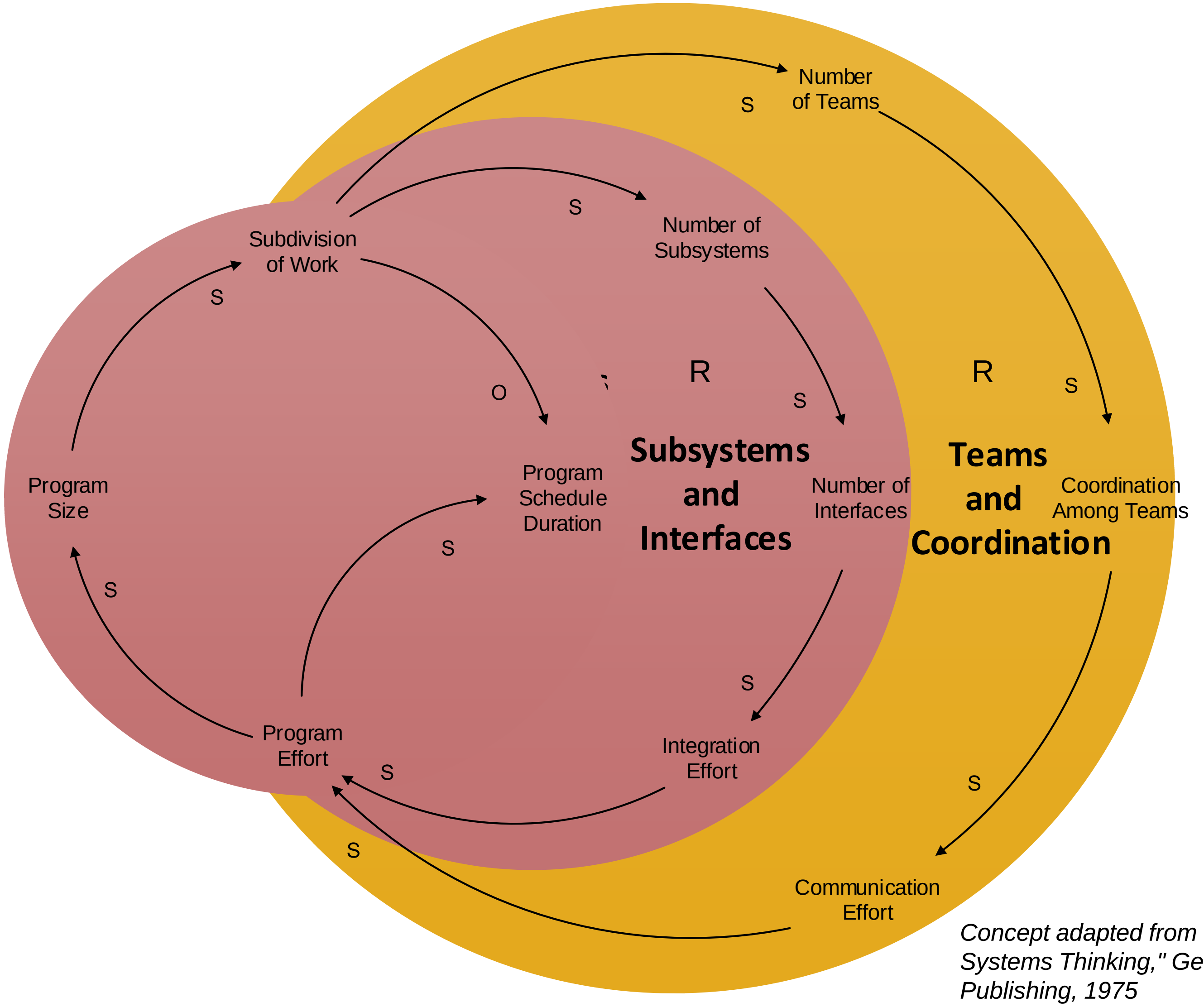
(Size of box represents number of findings)

Taxonomy:

- 22 top-level categories
- 114 categories total

Direction and Communication

“Divide and Conquer”

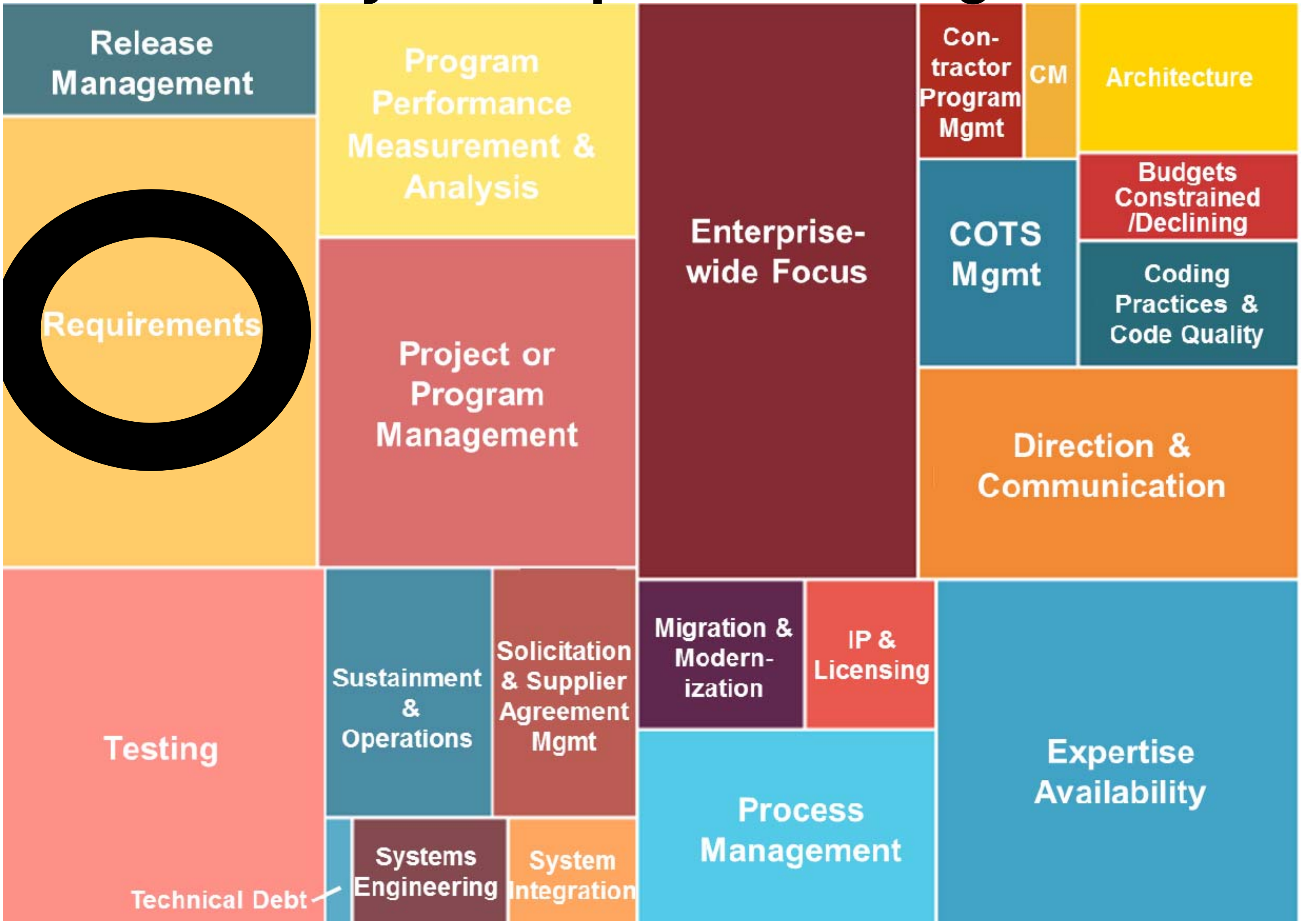


Very large programs require the partitioning or division of the work into tasks and teams that can then work in parallel. The consequence of this is that while it offers schedule benefits, it also creates coordination problems between the teams in terms of more communication channels as well as formal interfaces between the system components, making the integration effort grow nonlinearly. This undercuts and minimizes the originally intended benefits of the partitioning.

Concept adapted from "Quality Software Management, Vol. 1: Systems Thinking," Gerald M. Weinberg, Dorset House Publishing, 1975

Acquisition Dynamics

Taxonomy: 22 Top-Level Categories



449 findings in total

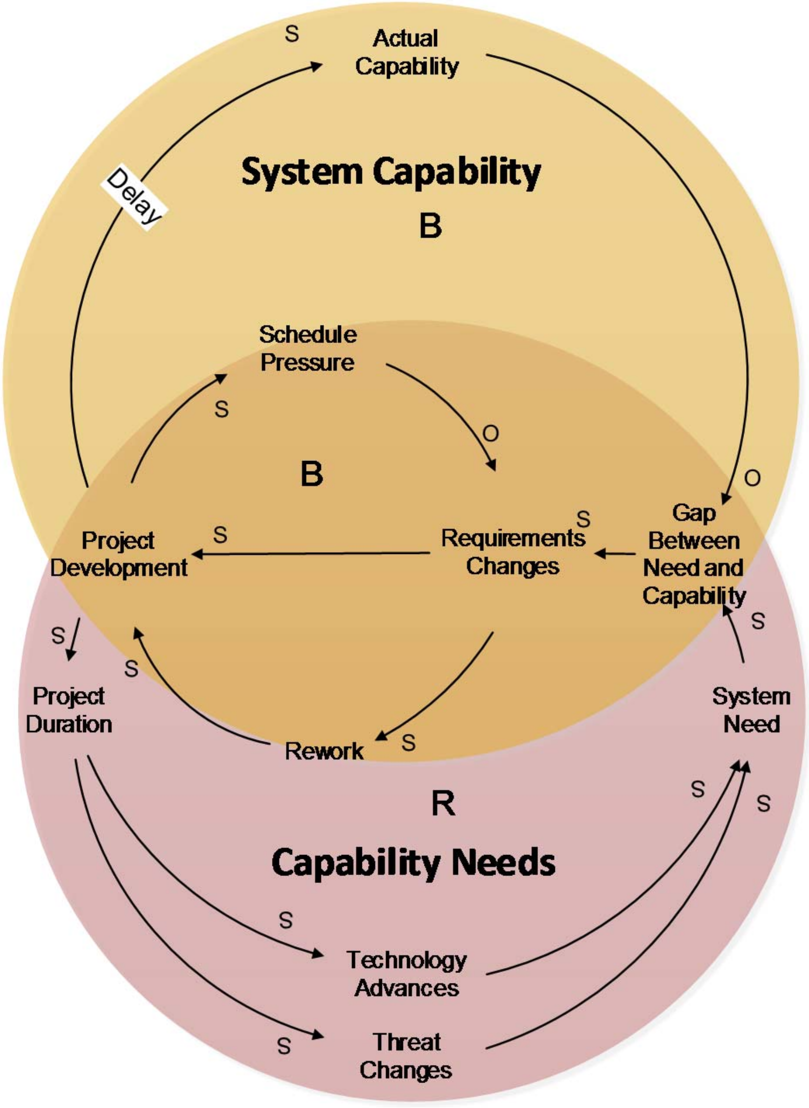
(Size of box represents number of findings)

Taxonomy:

- 22 top-level categories
- 114 categories total

Requirements

“Requirements Scope Creep”



The amount of Project Development drives longer Project Duration, during which Technology Advances and the Threat Changes, increasing the System Need, widening the Gap Between Need and Capability, and thus driving more Requirements Changes and Project Development through both new development and Rework. More Project Development increases the system’s Actual Capability, driving down the Gap Between Need and Capability. However, more Project Development also produces more Schedule Pressure, which reduces the number of Requirements Changes.

Analyzing the Results of 13
Acquisition Program Assessments

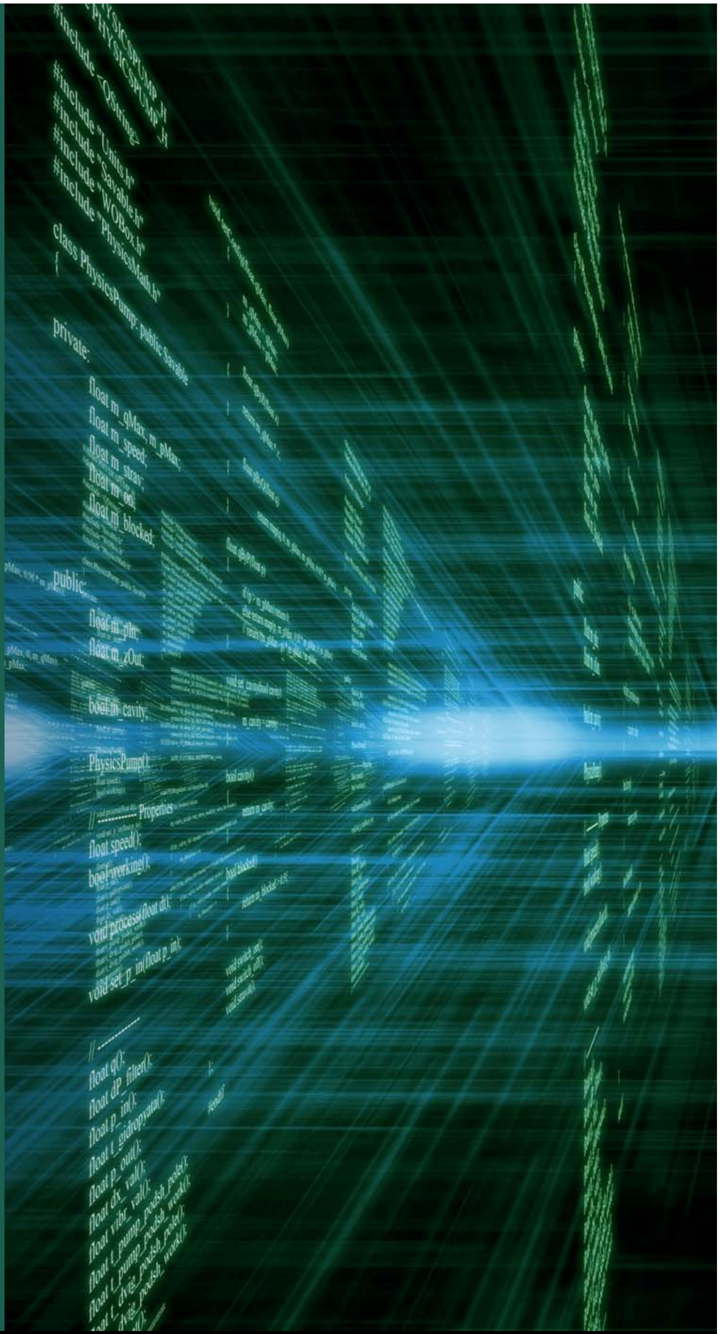
Breaking the Pattern

“You cannot apply a technological solution to a sociological problem.”

—Edwards’ Law

“A clever person solves a problem. A wise person avoids it.”

—Albert Einstein



Software Engineering Institute | Carnegie Mellon University

Breaking the Pattern

Managing the Acquisition Archetypes -1

If you're caught in a storm at sea, there may be little you can do to:

- Stop the storm, or even
- Get out of the storm

…But there *are*:

1. Things you can do *beforehand* to avoid it or minimize its impact, and
2. Things that you can do *during* the storm to help you weather it.

By showing the underlying *structure* of a dynamic, archetypes show where best to apply leverage to slow or stop it.

Breaking the Pattern

Managing the Acquisition Archetypes -2

Once a recurring behavior has been characterized in a causal loop diagram, here are some key techniques for managing it:

- Reverse the *direction* of the archetype
  - *Make negative dynamics positive ones by running them backwards*
- Slow down *unwanted* reinforcing loops
  - “*When you’re in a hole, stop digging*”
- Accelerate *desirable* reinforcing loops
- Change the *limit* that a balancing loop is stabilizing around
  - *Change the equilibrium value to something more acceptable*
- Shorten the duration of a delay
  - *Make it easier to manage by making cause→effect more evident*
- Find *leverage points* where a small effort can have a large effect
- Look for *misaligned incentives* and try to align them

Each systems archetype has specific interventions for addressing it.

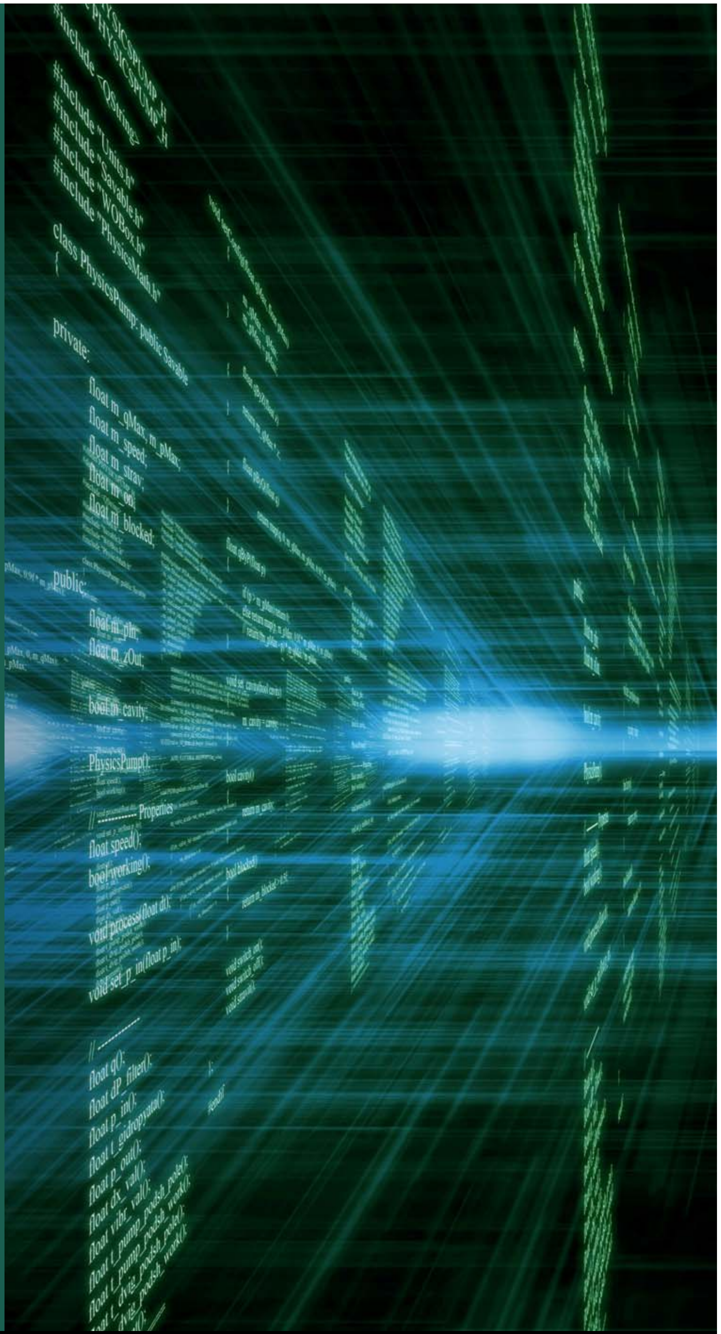
Knowing about these common dynamics is the best way to prevent them.

Analyzing the Results of 13
Acquisition Program Assessments

Solving Social Dilemmas

"Most [social traps]… are widely recognized to be genuine social problems, but the inclination to look at them as unrelated phenomena has obscured the possibility that similar sorts of solutions may be available across the board."

—John Cross and Melvin Guyer, Social Traps



Breaking the Pattern

Solving Social Dilemmas $_{-4}$

Motivational: Make people *want* to behave better

- ***Set Expectations/Reciprocity***: "If someone else does it, then I will, too"
- ***Awareness***: Raise awareness so that everyone knows how they should act
- ***Build Trust***: Let participants prove their trustworthiness so all are willing to cooperate
- ***Pulling Out***: Leaving the group if a partner defects sends a message to others

from Cross and Guyer, Social Traps, University of Michigan Press, 1980.

Breaking the Pattern

Solving Social Dilemmas -5

Strategic: Give people some *reason* to behave better

- ***Reputation***: Build public reputations based on past performance to boost confidence
- ***Order***: Avoid a precedence order for using the resource, encouraging equality of use

from Cross and Guyer, Social Traps, University of Michigan Press, 1980.

Solving Social Dilemmas -6

Structural: Change the rules so that people *must* behave better

- ***Authority***: Designate a leader/authority to regulate the use of the good
- ***Privatization***: Privatize the good so that each person pays for their use
- ***Rewards and Punishment***: Create clear rules, rewards, and penalties for behaviors
  - ***Reward the Group***: Reward people for group, rather than individual, success
  - ***Altruistic Punishment***: Allow participants to punish those who don't cooperate
- ***Assurance Contract***: Cooperate only if enough others also commit to do so
- ***Small Groups/Communities***: Small groups are more willing to "do it for their team"
- ***Exclusion Mechanism***: Find a way to exclude "free riders" from access
- ***Merging Free Riders***: Buy out the free riders so they have no incentive to free ride

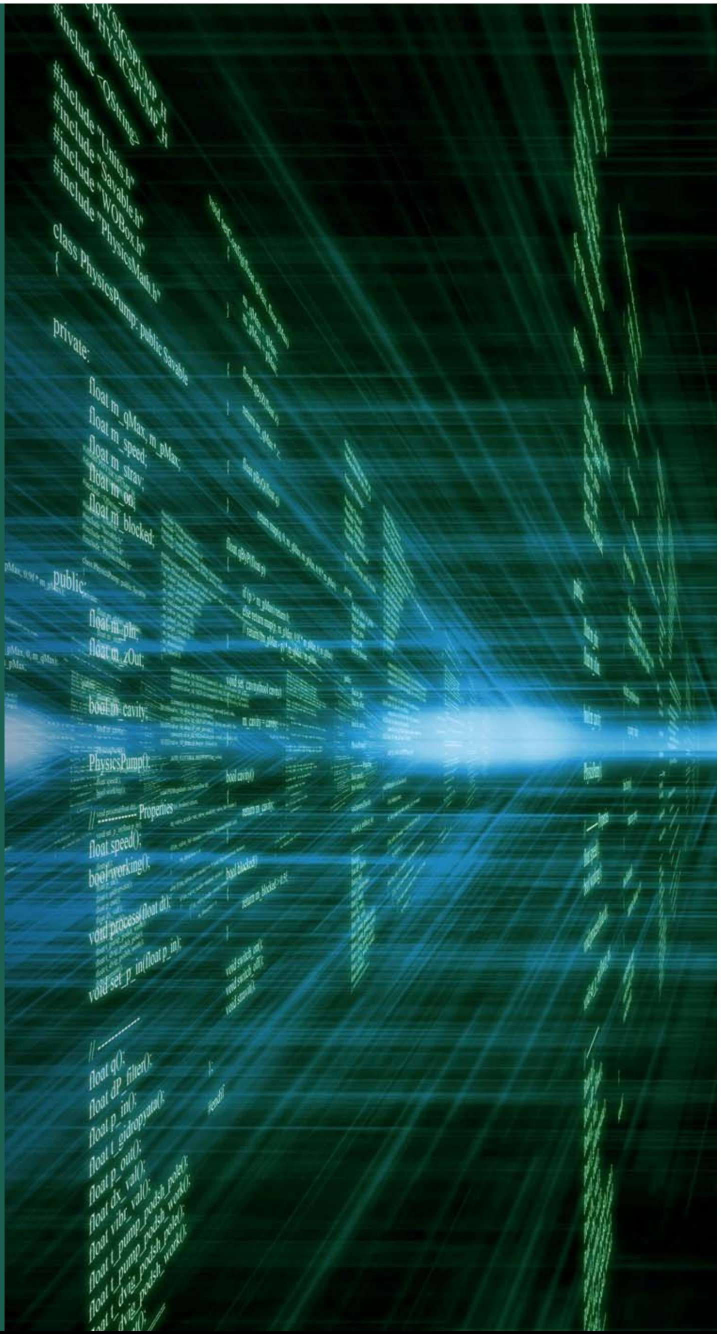
from Cross and Guyer, Social Traps, University of Michigan Press, 1980.

Analyzing the Results of 13
Acquisition Program Assessments

Summary and Conclusions

“If we want to change people's behavior, then we have to create circumstances in which people are likely to act virtuously… If we think about reforming people's character, we're engaged in a futile pursuit.”

—Randy Cohen, “The Ethicist,” New York Times



Summary and Conclusions

The Big Ideas -1

Lasting improvement to a system's behavior comes from changing the underlying system structure.

People are poor at controlling systems with large time delays between the cause and effect, because they obscure the connection between the two.

Diagrams of archetypes show the structure that lays beneath the visible problems, pointing out "leverage points" to help resolve them.

The ways people devise to *exploit* policies are *themselves* “emergent behaviors” that cannot be predicted from the rules of the system.

Understanding and changing the misaligned incentives at work beneath acquisition problems is key to improving program performance.

Summary and Conclusions

The Big Ideas -2

Acquisitions fail primarily for non-technical reasons

- *Organizational, management, and cultural issues dominate*
- *“Technology has gotten ahead of our organizational and command capabilities in many cases”*

Misaligned incentives drive counter-productive behaviors

- *Programs put their own good ahead of other programs*
- *Programs put their good ahead of their service’s good*
- *Programs place short-term considerations ahead of longer-term ones*

Understanding the problem is the first step toward solving it

- *If these problems were easy to solve, they wouldn’t still be plaguing us*
- *There is no simple boilerplate answer—but there are solutions*

Distribution Statement A: Approved for Public Release; Distribution is Unlimited

Summary and Conclusions

The Big Ideas -3

Seemingly simple systems produce unexpectedly complex behaviors

- *New behaviors can emerge from interactions among components*

Small changes in initial inputs drive big changes in system results

- *Minor incidents can escalate into major catastrophes*
  - *PMO vs. Contractor Hostility (Accidental Adversaries)*
  - *Robbing Peter to Pay Paul (Success to the Successful)*

Assumptions contribute to failing acquisitions

- *Assumptions about others predispose you to behaving in certain ways*
- *Articulate underlying assumptions that contribute to misaligned incentives*

Lack of trust can degenerate into turf wars and a “death spiral”

- *If “individual/team gain” trumps the “program’s good,” bad outcomes result*

Summary and Conclusions

For Additional Information

SEI Report: “*The Evolution of a Science Project: A Preliminary System Dynamics Model of a Recurring Software-Reliant Acquisition Behavior*”

SEI Report: “*Success in Acquisition: Using Archetypes to Beat the Odds*”

SEI Blog: “*Themes Across Acquisition Programs”: Parts 1-4*

Website: http://www.sei.cmu.edu/acquisition/research/archetypes.cfm

Download all twelve:

- PMO vs. Contractor Hostility
- Underbidding the Contract
- Everything for Everybody
- The Bow Wave Effect
- Brooks' Law
- Firefighting
- "Happy Path" Testing
- Longer Begets Bigger
- Shooting the Messenger
- Feeding the Sacred Cow
- Staff Burnout and Turnover
- Robbing Peter to Pay Paul

Acquisition Archetypes
Firefighting
Changing Counterproductive Behaviors in Real Acquisitions
PMO versus Contractor Hostility
Sacred Cow
Effect

For More Information

William E. Novak
wen@sei.cmu.edu
412.268.5519

Dr. Forrest J. Shull
fjshull@sei.cmu.edu
703.247.1372

Thank You!