# Using System Archetypes to Identify Failure Patterns in Acquisition

Diane Gibson
Linda Levine, PhD
William E. Novak

May 2, 2006

Carnegie Mellon
Software Engineering Institute

# Overview

- Introduction to Systems Thinking

Applying Archetypes to Acquisition:
The Far Orbit Satellite System (FOSS)

Conclusions and Future Directions

Carnegie Mellon
Software Engineering Institute

# What is "Systems Thinking"?

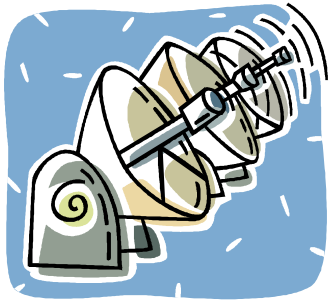
- A new perspective which provides an understanding that comes from focusing on the big picture and the interactions internal and external to it instead of dividing up the picture into smaller elements.

For example -- if you look at the body of an aardvark, you see one picture if you kill it, cut it up and examine the parts; you see another if you look at the animal as one entity, and examine how it interacts with different levels of the environment in which it lives (or if you study the interactions of the organs which make up the system called "aardvark".)

- it involves thinking in 'circles' rather than straight lines
- cause and effect are interactive; what is a cause one time is an effect another

Carnegie Mellon
Software Engineering Institute

# What is a System?

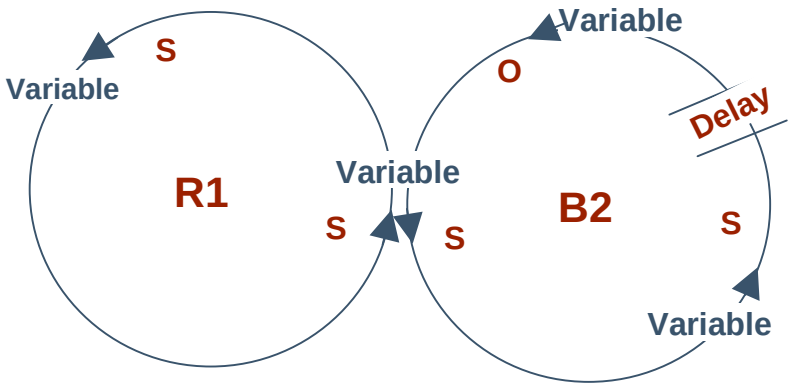


System = a collection of elements where

- the whole is bigger than the sum of the parts
- everything is interconnected & effects everything else
- there is a purpose & the elements are arranged meet this purpose
- the interactions of elements are characterized by reciprocal flows of influence
- this is also what causes the system to change
- the same action may have different effects, different actions may have the same effects, & delays often lead to unexpected outcomes
- cause & effect may be far apart in time & space

# Causal Loop Diagrams - Language of Systems

- Consist of lines with arrows, drawn in circular format, to show cause-effect (influence) relationships
- The direction of the relationship is labeled (S)ame or (O)pposite, based on the direction of the causation.
- Loops are labeled either
  - (R)einforcing - to change by making stronger/weaker; created with an even # of opposites.
  - or (B)alancing - equilibrium achieved between contrasting or opposing elements; created with an odd # of opposites.
- Time delays are important.
- Additional loops can be added to show side effects or other consequences.



Carnegie Mellon
Software Engineering Institute

# Systems Archetypes

- Story, diagram, template and/or graph
- that describes and depicts specific patterns
- which reveal the underlying system structure and associated problems and mental models
- which facilitates addressing the underlying causes of behavior
- which are common across many organizations

Carnegie Mellon
Software Engineering Institute

# Using Archetypes

- Viewpoints (Lenses)
- Examine problem structure (Patterns)
- Explain dynamic events
- Forecast potential behaviors
- Make changes to a system
- Present information about problems & solutions

Carnegie Mellon
Software Engineering Institute

# Different Archetypes

- Shifting the Burden
- Drifting Goals
- Limits to Success/Growth

Carnegie Mellon
Software Engineering Institute

# Shifting the Burden - Story

When a quick fix is applied to a problem, and it seems to work, that is the end of further investigation.

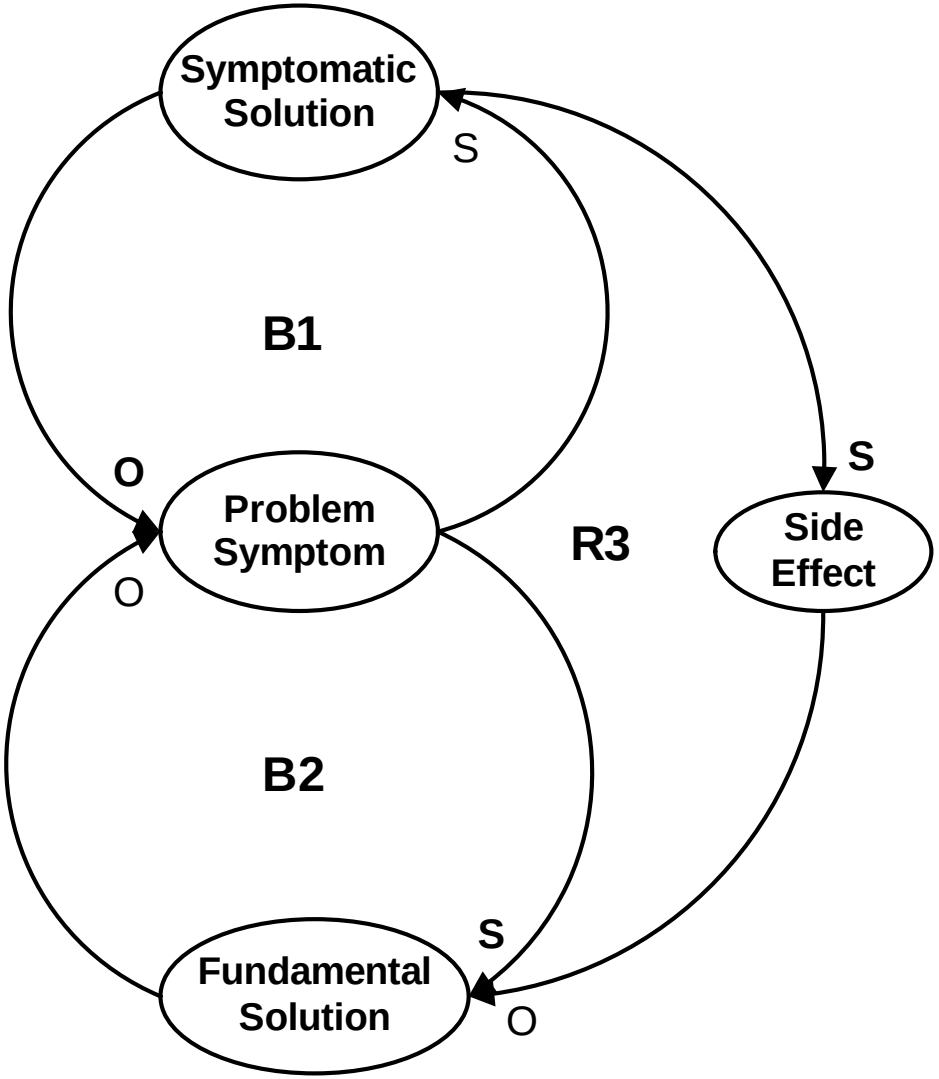
THE *REAL* PROBLEM...

However, the quick fix

- leads to side effects, which may not be recognized as related to the original problem;
- consumes resources, reducing the ability of the organization to address other causes and ultimately, to implement a fundamental solution;
- and since the fundamental problem isn't addressed, it returns, often in a more virulent form.

# Shifting The Burden

## Causal Loop Diagram



Carnegie Mellon
Software Engineering Institute

# Shifting The Burden

Comments like the following should give you a clue that you may be experiencing a shifting the burden pattern..

- Just once more won't do any harm.
- The next time this happens we'll look at it afresh.
- I wish we could stop this, it's not doing any good.
- When this emergency ends, we'll return to our normal standards.
- I know it doesn't help in the long run, but what can we do?
- It's not my responsibility!

Carnegie Mellon
Software Engineering Institute

# Shifting the Burden – Interventions

- Identify problem symptoms, side effects and the quick fixes.
- Focus on the fundamental solution; maintain the symptomatic solution while working on a fundamental solution.
- Examine the impact of solutions on different groups in the organization. Gather multiple viewpoints to distinguish between the symptomatic and fundamental solutions.
- With an addiction pattern, discover what in the basic structure of the system contributed to making the fix an addiction.
- Use the archetype to explore potential side effects of any proposed solution.

Carnegie Mellon
Software Engineering Institute

# Drifting Goals – Story

A person or organization seeks to achieve a goal. But, it will take a lot of time and money to do so.
So, they decided to reduce the goal, just a little...
Voila! The difference quickly goes away with little delay or cost!

## THE *REAL* PROBLEM*?*

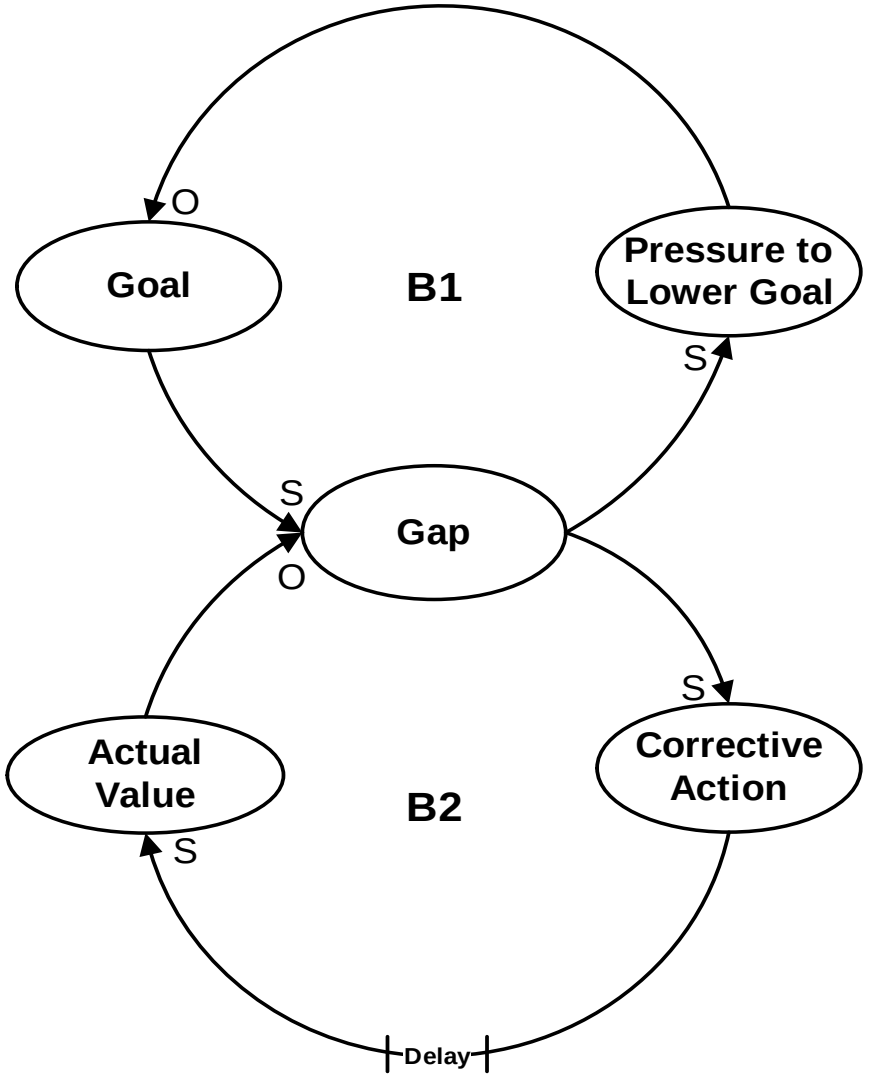
Reducing the goal may continue and become a habit.

Someone must acknowledge that a problem exists or the goal may continue to erode without recognition.

NOTE: sometimes the goal may actually be too high. You may need to take the time to evaluate.

# Drifting Goals

## Causal Loop Diagram



Carnegie Mellon
Software Engineering Institute

# Limits to Success – Story

People engage in certain activities, leading to success. As efforts increase, success increases. The growth engine works.
After time, growth slows down. The reaction – redouble efforts, and do more of the same.
Paradoxically, as you push harder, the system pushes back.

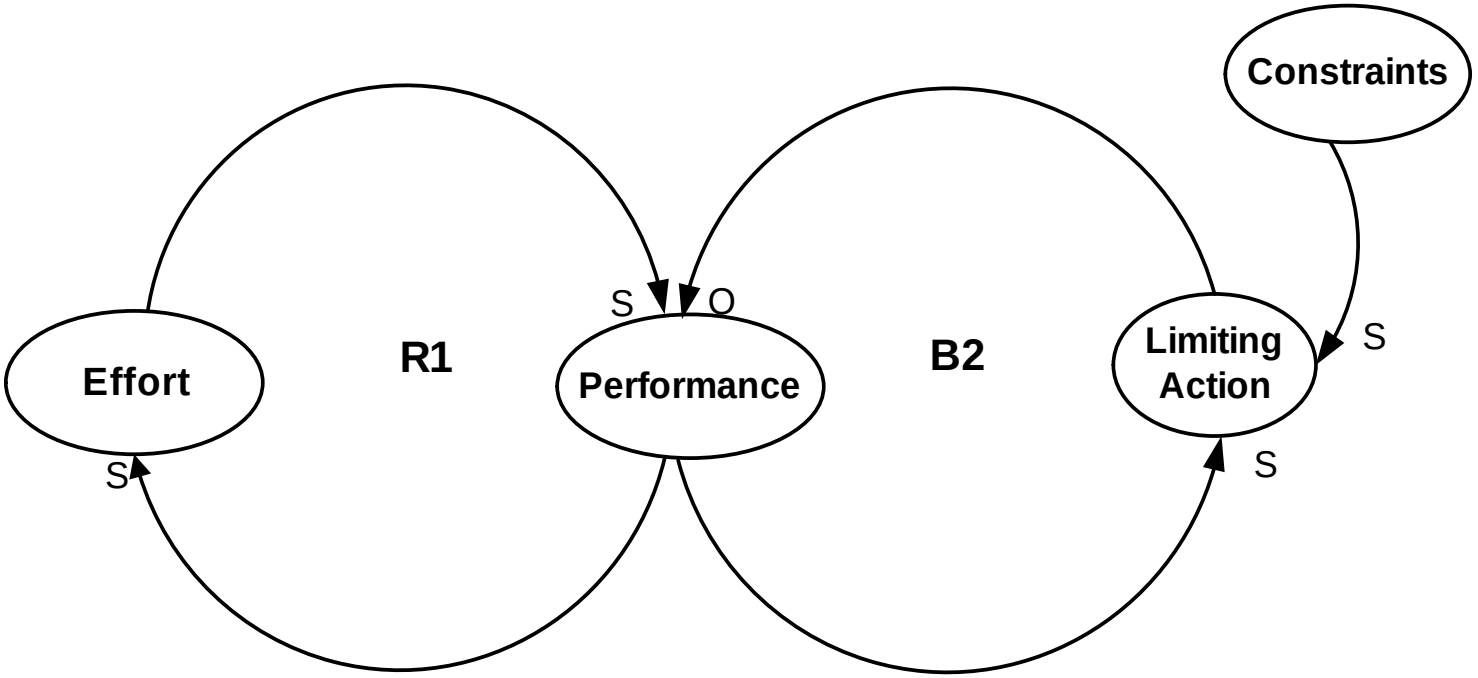
THE *REAL* PROBLEM...

Over time, limiting forces, intimately connected to the growth engine, come into play. As they begin to dominate, growth and performance improvements stabilize or decline.
Doing what initially worked creates its opposite. To restart growth, a way must be found to deal with the limiting forces.

Carnegie Mellon
Software Engineering Institute

# Limits to Success –

## Causal Loop Diagram



Carnegie Mellon
Software Engineering Institute

# Overview

Introduction to Systems Thinking

Applying Archetypes to Acquisition:
The Far Orbit Satellite System (FOSS)

Conclusions and Future Directions

# FOSS Program Case Study -1

## Far Orbit Satellite System

Satellite space surveillance ground segment – cost of $2.1B ($1.8B with a $350M award fee)

**Nov '96** A Cost-Plus EMD contract awarded to Genadyne, teamed with Aerocorp.

Increment 1 was due in September 1999, Increment 2 was due in October 2002, and Increment 3 due in 2009. The 24 year lifecycle cost was estimated at $10B.

Per acquisition reform, the program embraced TSPR, used performance-based requirements, had minimal government SPO staff (since contractor carried the risk), and relied on contractor data and metrics.

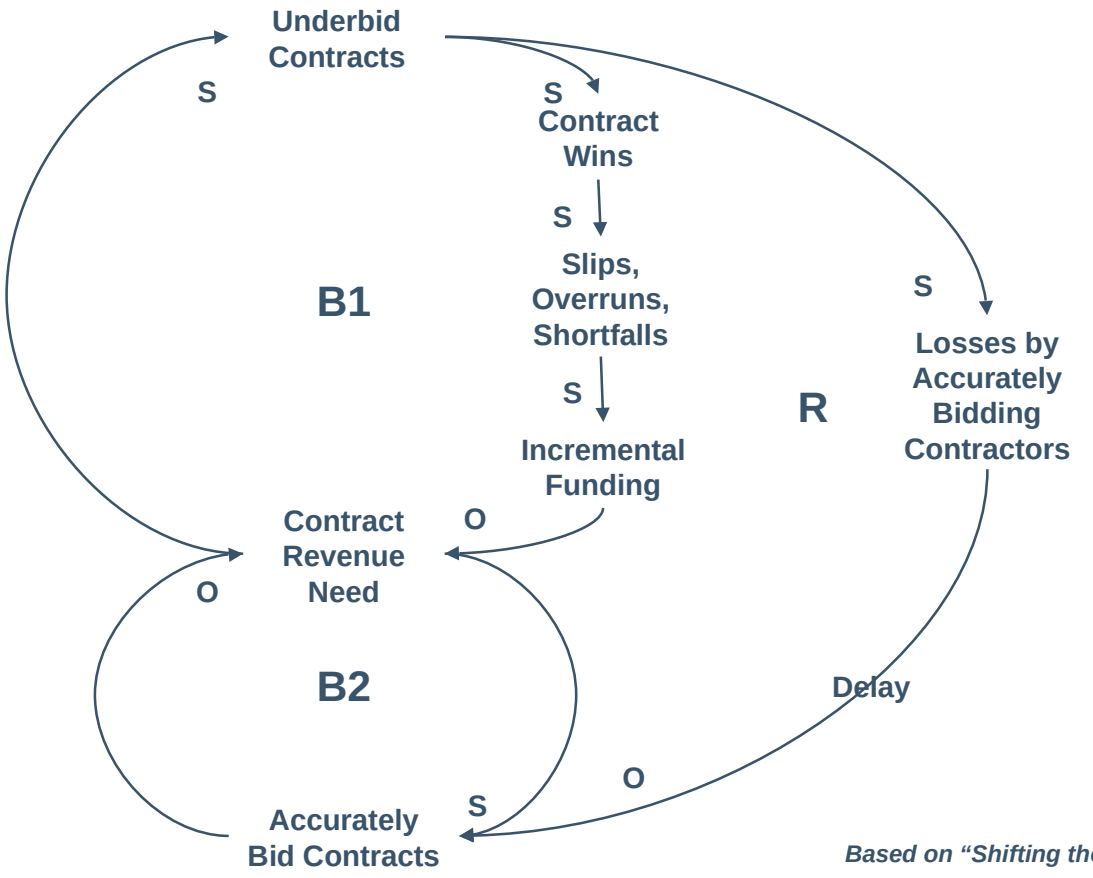
The program planned to use spiral development, software reuse, and open systems architecture for modular updates & COTS use.

# Underbidding the Contract

*"The SPO and the Office of the Secretary of Defense (OSD) had estimated FOSS EMD costs to be $3.2 billion. The contactor had bid $1.8 billion with a $347 million award fee pool for a total contract award of $2.1 billion. The large gap between the Government estimate and the contractor's bid raised some concern…"*

*"The government model estimated that the software could be built at X rate and the contractor maintained that they could build it at 3X rate. This was not resolved at source selection. Since the TSPR paradigm was in full ascendancy, it was decided to let the contractor do the work with their assumption."*

# Underbidding the Contract - Diagram




*Based on "Shifting the Burden"*

# FOSS Program Case Study -2

Aerocorp was given $105M for Increment 1 work.

**Jul '97** A COTS vendor went out of business forcing a replan, but the government still paid the front-loaded award fees.

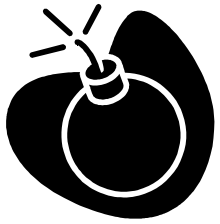
Increment 1 was on a very tight schedule. It was assessed to be at high schedule risk, although only low technical risk.

Carnegie Mellon
Software Engineering Institute

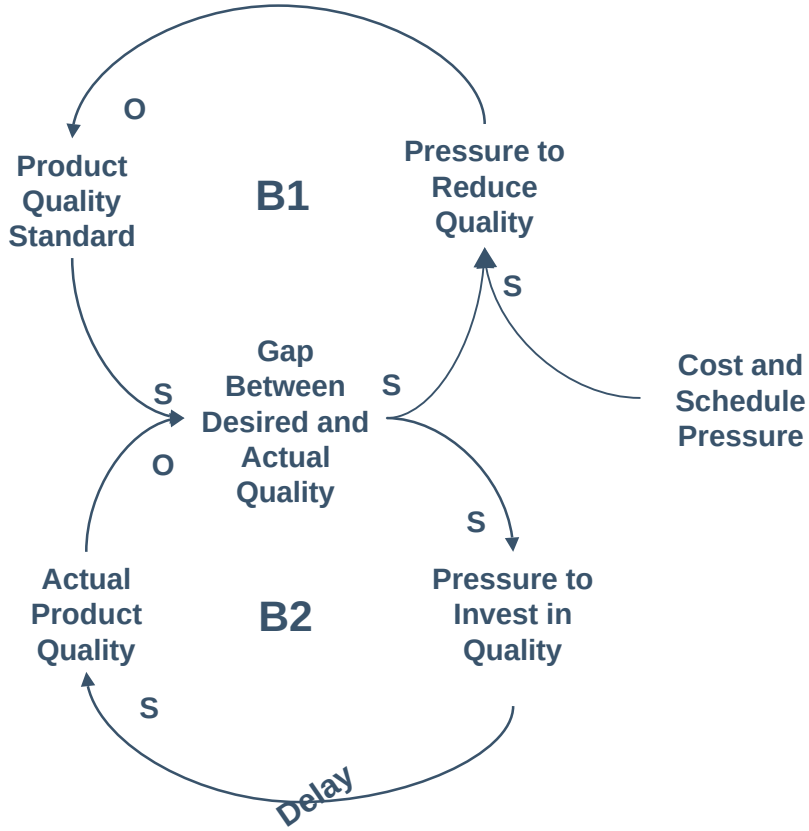
# Sacrifice Quality

*"The government model estimated that the software could be built at X rate and the contractor maintained that they could build it at 3X rate... The metric was lines of code. Unfortunately, lines of code does not mean lines of good code… they had a quota they were reporting on the lines of code they had achieved. At the lower levels, the engineers were writing code, but all of the quality checks were falling off the table. Things like peer code checks were not happening."*

*"Some SPO technical staff reviewed contractor software tests and became aware of dubious practices of 'passing' test milestones even though the tests had revealed significant software problems. Basically, the test failures were noted and the corrections were deferred for future tests."*



# Sacrifice Quality - Diagram



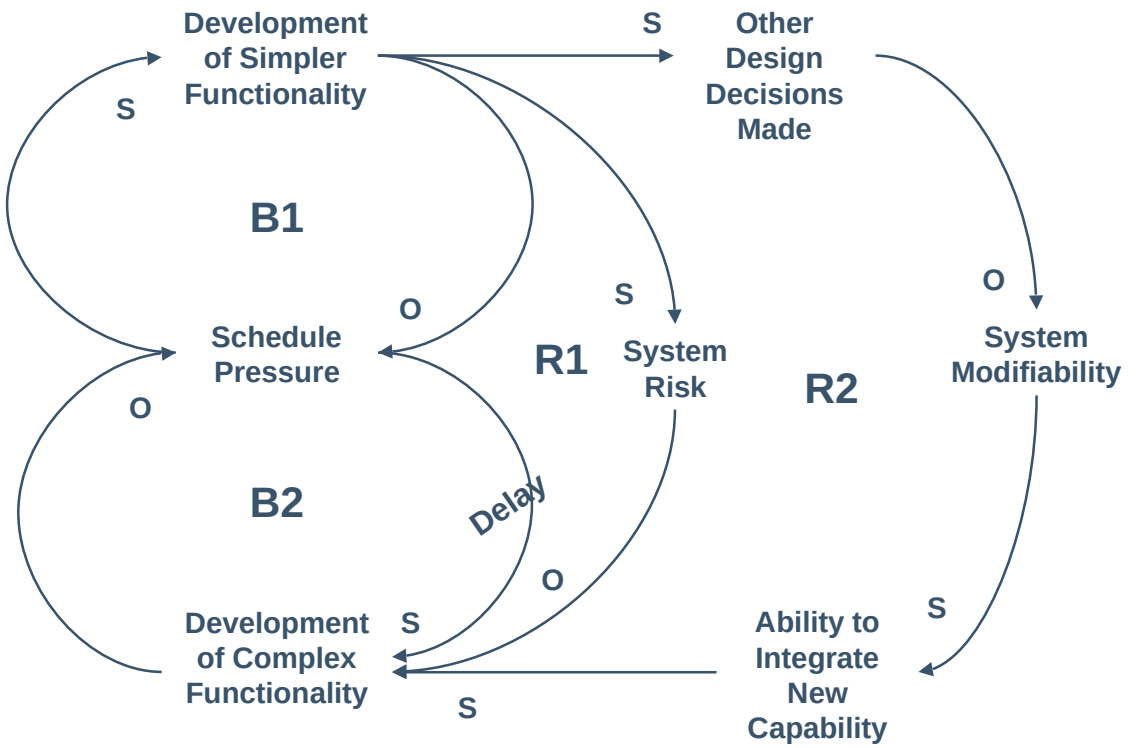

*Based on "Drifting Goals"*

Carnegie Mellon
Software Engineering Institute

# The Bow Wave Effect

*"Another Increment 1 software issue involved spiral development. Aerocorp software engineers would claim that they were meeting milestone events, even though content was being shifted from one spiral to the next. Aerocorp said that it was appropriate under the spiral paradigm to 'pick the low-hanging fruit first' and defer difficult requirements. They said this practice insured successive spirals could be implemented and tested promptly, and that valuable test results could be fed back into the next spiral."*

# The Bow Wave Effect - Diagram




*Based on "Shifting the Burden"*

Carnegie Mellon
Software Engineering Institute

# FOSS Program Case Study -3

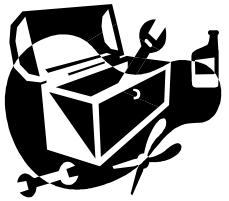
**Jan ‘99** The problems with passing tests continued to increase, but the contractor said they were normal.

**Jun ‘99** The contractor found that the reusable software couldn’t be used.

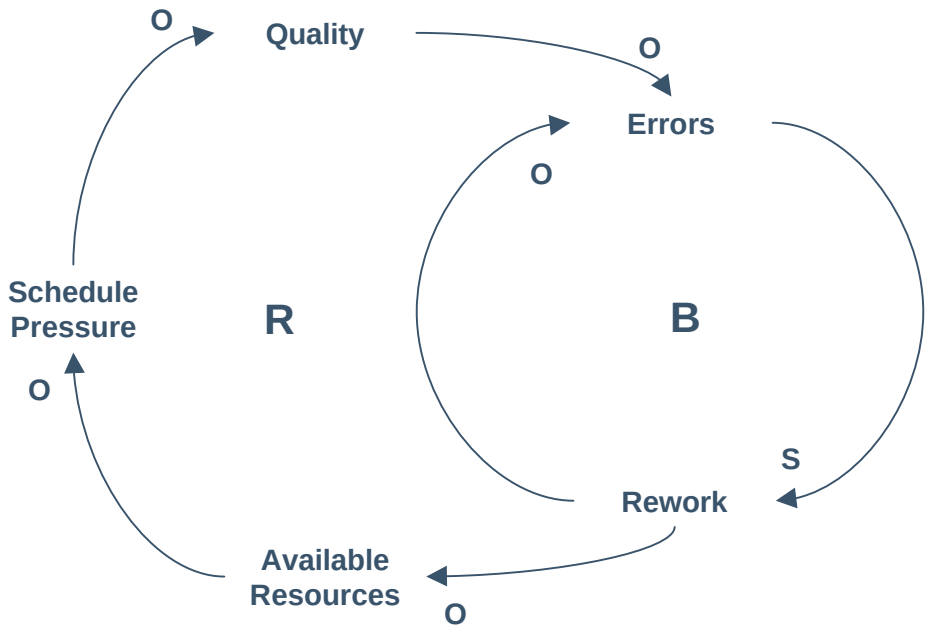
Carnegie Mellon
Software Engineering Institute

# Firefighting

*"When Aerocorp began system testing in the summer of 1999, they plunged into a persistent test-fix-test cycle. It was impossible to verify system performance because the system would not run long enough before crashing. Instead, the testing became part of continued system development. The system architecture would not support performance requirements; the timing of system events was off. …it had both software integration and hardware-software integration problems."*



# Firefighting - Diagram




*"Firefighting" concept from "Past the Tipping Point"*

Carnegie Mellon
Software Engineering Institute

# FOSS Program Case Study -4

Increment 1, due in September 1999, was far from done due to architecture, design, and process issues.

**Aug '99** The SPD proposed declaring a breach against the 1999 delivery, but the PEO agreed with the contractor saying, "It could still happen"—but Increment 1 failed repeatedly during OT&E testing.

**Oct '99** FOSS declared an Acquisition Program Baseline (APB) breach. They brought in a Management Assessment Team (MAT) for recovery planning. The MAT concluded that management and development processes were inadequate, Increment 1 had high technical risk, the design should be reengineered, and the program would slip 2 years.

**Jul '00** The SPD restructured FOSS to MAT recommendations. They assessed the contractor $35M for late delivery. The SPO assigned a colonel to closely monitor Increment 1.

Carnegie Mellon
Software Engineering Institute

# FOSS Program Case Study -5

**Jul '00** In IBR the contractor estimated $500M funding shortfall due to MAT restructuring, replanning, improved testing. The SPD got that reduced to a more acceptable $277M, and the AF CAIG validated that.

**Nov '00** OSD CAIG estimated a $510M shortfall, not $277M, so OSD increased funding by $484M.

**May '01** A payload subsystem failure occurred requiring significant redesign, and contractor requested $125M more.

**Jul '01** The new SPD calculated an EAC of $318M over the current budget and a 2-year slip.

Carnegie Mellon
Software Engineering Institute

# Don’t Bring Me Bad News

*“The Genadyne VP [said they] would need $125 million more in FY02… Disputing the… Genadyne Vice President, Blake told Mr. Collins that with aggressive cost controls, the $45 million would suffice. Blake then retired.”*

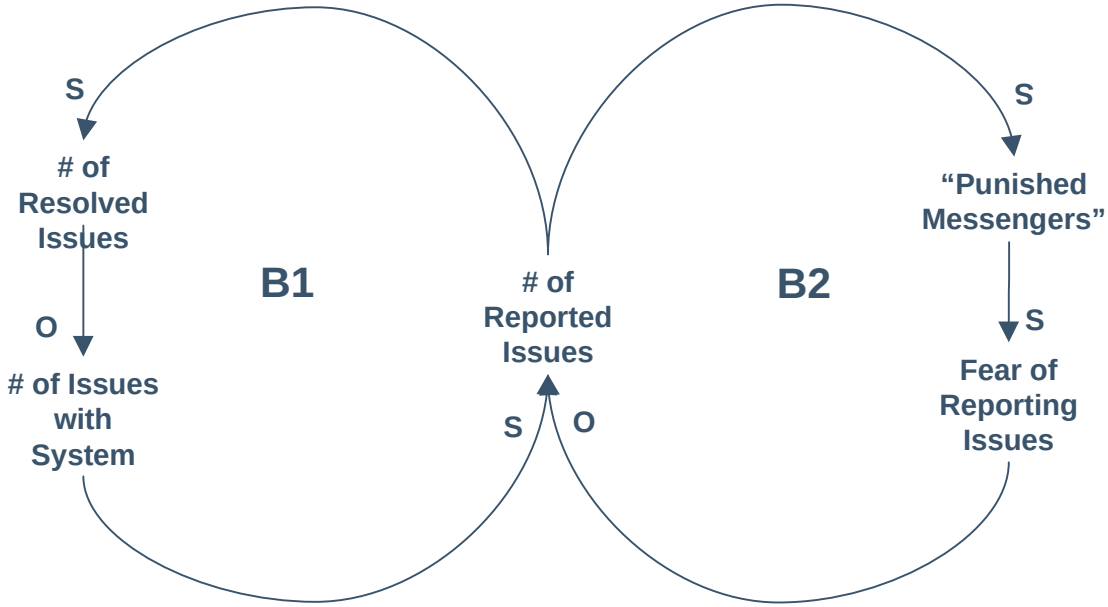
*“Broderick passed the bad news informally up his chain of command. His superiors warned him that cost overruns of the scale he was suggesting might cause program cancellation. They urged caution and deliberation…*

*“Col. Broderick wanted to get the bad news out, and again told the Air Force leadership in the Pentagon that FOSS was at least $1 billion short. He figured this would be a ‘significant emotional event’ for Mr. Collins...”*

*“Later that same day, Mr. Collins was in Lt Gen Ashton’s office and placed a speakerphone call to Col Broderick. Col Broderick recalled their voices as very stressed. Again the issue was about reducing the bill. Broderick said, ‘I can’t do that.’”*

Carnegie Mellon
Software Engineering Institute

# Don’t Bring Me Bad News - Diagram



Carnegie Mellon
Software Engineering Institute

# FOSS Program Case Study -6

| | |
|---|---|
| **Oct '01** | SPD met with with the Air Force Deputy Assistant Secretary of Acquisition & Management and the contractor VP about the shortfall, and the VP said the bad news was premature, but had no alternate estimate. The Deputy Assistant Secretary told the Air Force leadership of the shortfall. |
| **Nov '01** | FOSS initiated a Nunn-McCurdy breach acknowledging a more than 25% cost overrun. |
| **Dec '01** | FOSS Increment 1 finally passed OT&E, 2 years late. |

Carnegie Mellon
Software Engineering Institute

# Overview

Introduction to Systems Thinking

Applying Archetypes to Acquisition:
The Far Orbit Satellite System (FOSS)

Conclusions and Future Directions

Carnegie Mellon
Software Engineering Institute

# Systems Thoughts

*"Humankind has succeeded over time in conquering the physical world and in developing scientific knowledge by adopting an analytical method to understand problems ... this sort of linear and mechanistic thinking is becoming increasingly ineffective to address modern problems... because, today, most important issues are interrelated in ways that defy linear causation*."

*"Systems thinking allows people to make their understanding of social systems explicit and improve them in the same way that people can use engineering principles to make explicit and improve their understanding of mechanical systems."*

Daniel Aronson

Carnegie Mellon
Software Engineering Institute

# Observations -1

- Not all problems fit an archetype
- Non-linear nature of systems thinking
- People in the same structure tend to behave in similar ways

Carnegie Mellon
Software Engineering Institute

# Observations -2

Short-Term Thinking – Fixing Symptoms instead of Problems

- Penchant for the “quick fix” and “low-hanging fruit”
- Unwillingness to invest in long-term solutions addressing root cause

The Role of Intent and Motive

- Intent isn’t relevant—only *behavior* matters

Cross-Program Archetypes and Dynamic Games

- Learning from past actions and outcomes of analogous situations

Carnegie Mellon
Software Engineering Institute

# Next Steps/Future Directions

Integration of Archetypes
- Link related archetypes to build more complete models

System Dynamics Modelling
- Inputs to system dynamics simulations of acquisition

High-leverage Interventions
- Provide early warning of developing failure patterns

“Big Picture” Thinking
- Improve decision-making by avoiding oversimplification

Carnegie Mellon
Software Engineering Institute

# Additional Information

System Archetype Basics: From Story to Structure, Daniel H. Kim and Virginia Anderson, Pegasus Communications, Waltham, MA, 1998.

“Large-Scale Projects as Complex Systems: Managing ‘Scope Creep,’” Andrea Shapiro and Carol Lorenz, The Systems Thinker, vol. 11, num. 1, February 2000, Pegasus Communications.

Far Orbit Satellite System (FOSS), Case Study, Dr. Christopher Roman, Defense Acquisition University (DAU), Ira Monarch, SEI.

Defense Acquisition Performance Assessment Report, LtGen. Ronald Kadish, USAF (Ret) Panel Chairman, Defense Acquisition Performance Assessment Project Assessment Panel, January 2006.

Carnegie Mellon
Software Engineering Institute

# Contact Information

Diane Gibson
Senior Member of Technical Staff
Software Engineering Process Management Program
(412) 268-8993
dlg@sei.cmu.edu

Bill Novak
Senior Member of Technical Staff
Acquisition Support Program
(412) 268-5519
wen@sei.cmu.edu

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890