Modeling the Evolution of a Science Project in Software-Reliant System Acquisition Programs

Andrew P. Moore
William E. Novak
Software Engineering Institute
Carnegie Mellon University

International Conference of the System Dynamics Society

July 24, 2013

Acquisition Dynamics

The Evolution of a Science Project

9. Warfighters wait years for a new system to be built from scratch.

8. New versions of the system can't be deployed with needed capability, robustness, and performance.

7. New program office unwilling to discard prototype code due to field deployment pressures.

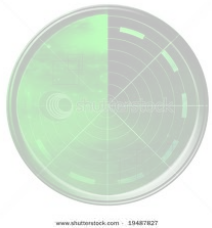
6. Project infrastructure, processes, & staff not able to scale up to production development.

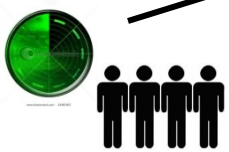
5. As system grows, poor architecture, documentation, & code quality cause poor reliability, performance, & usability.

3. Warfighters and field commanders demand more capability, broader deployment, faster response.

2. Prototype is deployed on small scale, and is well received.

1. Project begins as small informal effort to build prototype & prove concept.





4. Project staff is diverted to field support, so development progress slows.

This scenario aggregates five SEI software-reliant system acquisition ITAs conducted in 2006-2009.

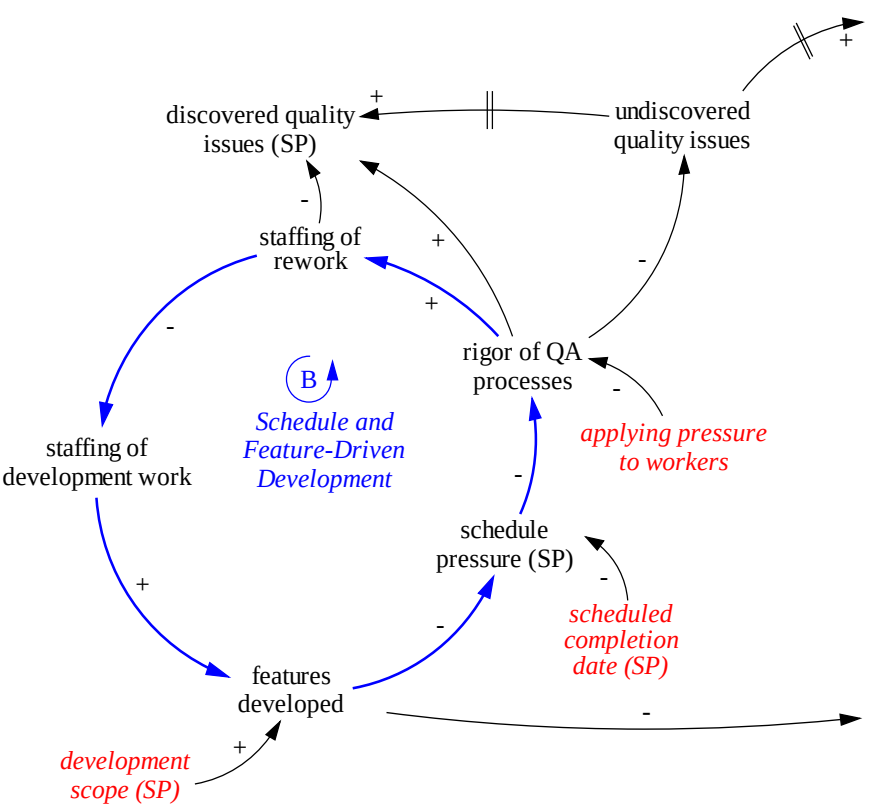
Basis for Modeling: Independent Technical Assessments

ITA: Detailed examination of challenged programs with interviews, document reviews, and code analysis

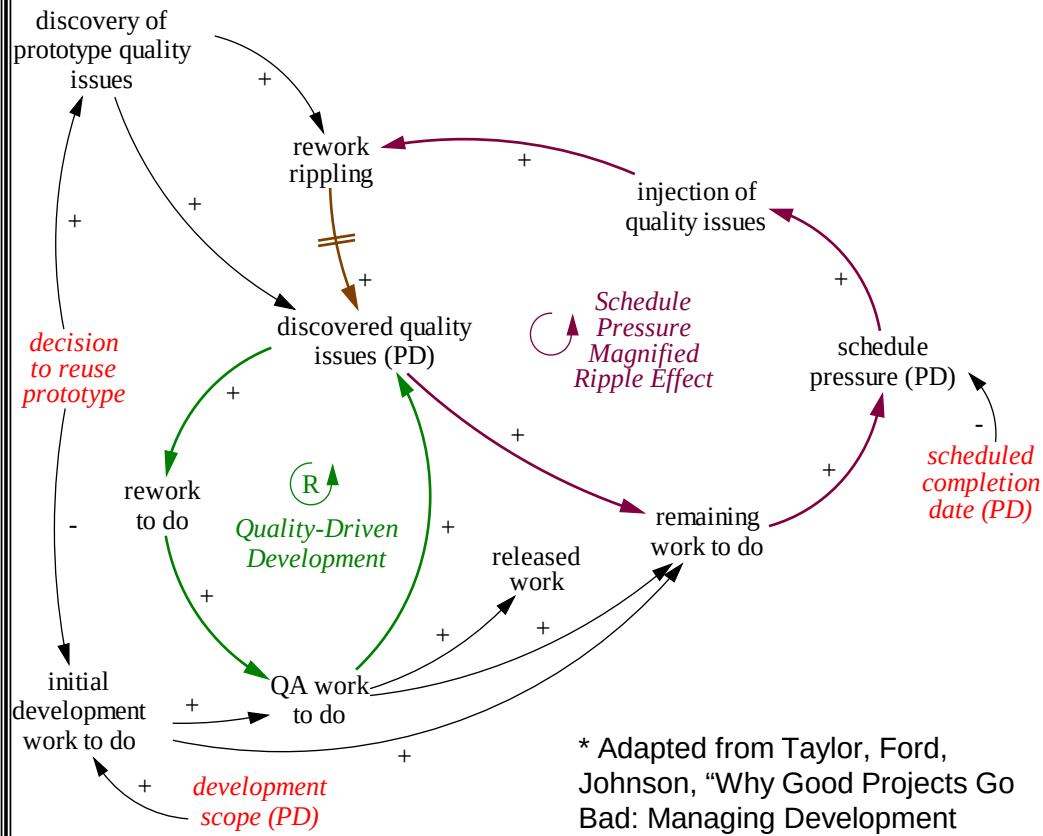
> "What they did at first was a proof of concept, a quick and dirty prototype, and when they tried to scale it up, there were indications that it might not be possible…"
>
> —*Acquisition Program Lead*

The Evolution of a Science Project

Science Project (SP) Sector



Production Development (PD) Sector *

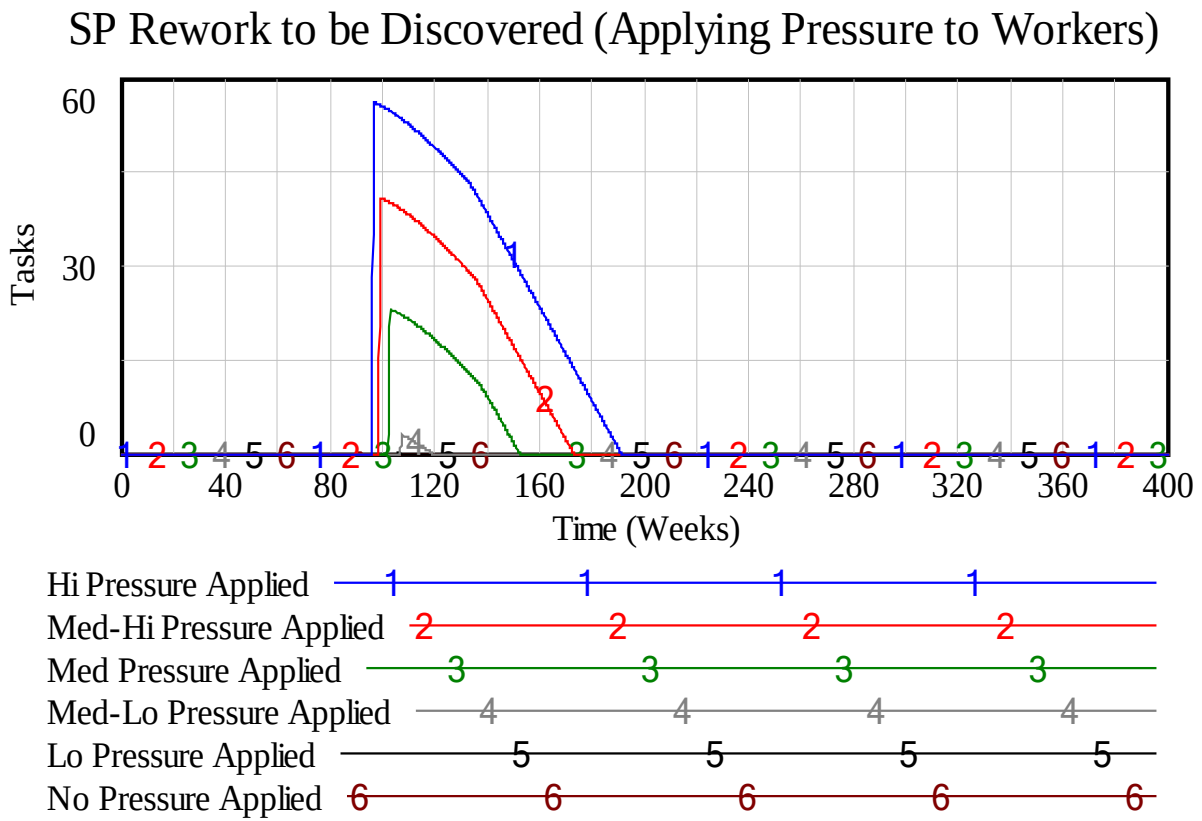


* Adapted from Taylor, Ford, Johnson, “Why Good Projects Go Bad: Managing Development Projects Near Tipping Points,” ICSD 2005, Boston.

Key Preliminary Findings

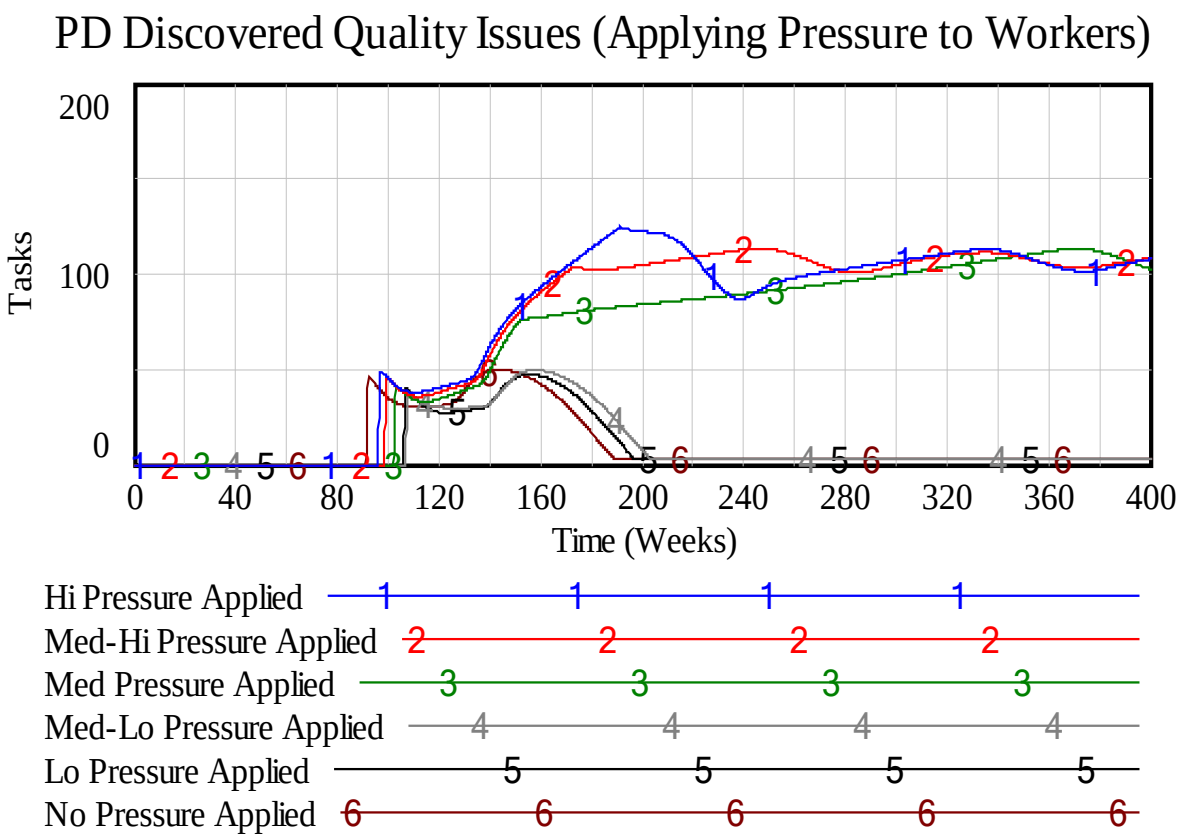
Assumption

Applying pressure to workers developing SP results in undiscovered rework



Key Preliminary Findings -1

High pressure, or moderate pressure for long periods, can lead to a “tipping point”



The Tipping Point in Evolution of a Science Project

- Accumulating rework creates a dangerous feedback dynamic
- "Firefighting" due to rework is a key underlying element
- Key drivers in reaching the "tipping point" are:
  a) pressure on developers
  b) emphasis on schedule and features vs. quality
  c) timing of the transition from science project to production development
  d) degree of "ripple effect"

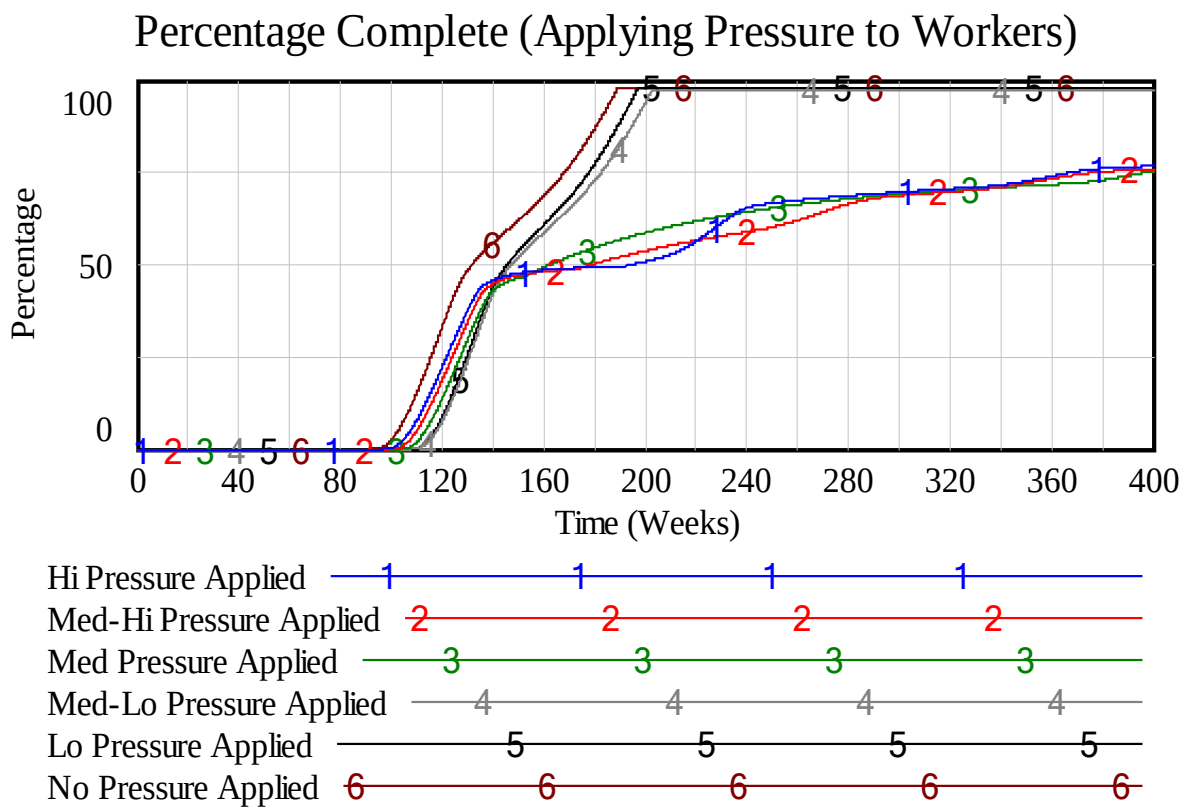
Key Preliminary Findings -2

Placing modest pressure on developers for limited periods shortens schedule

- VenSim optimization shows that placing pressure at a low level is optimal with respect to reducing project duration
- By allowing periods of pressure, followed by periods of relaxation, the program might:
  - Limit worker burnout
  - Perform better regarding schedule

Key Preliminary Findings -3

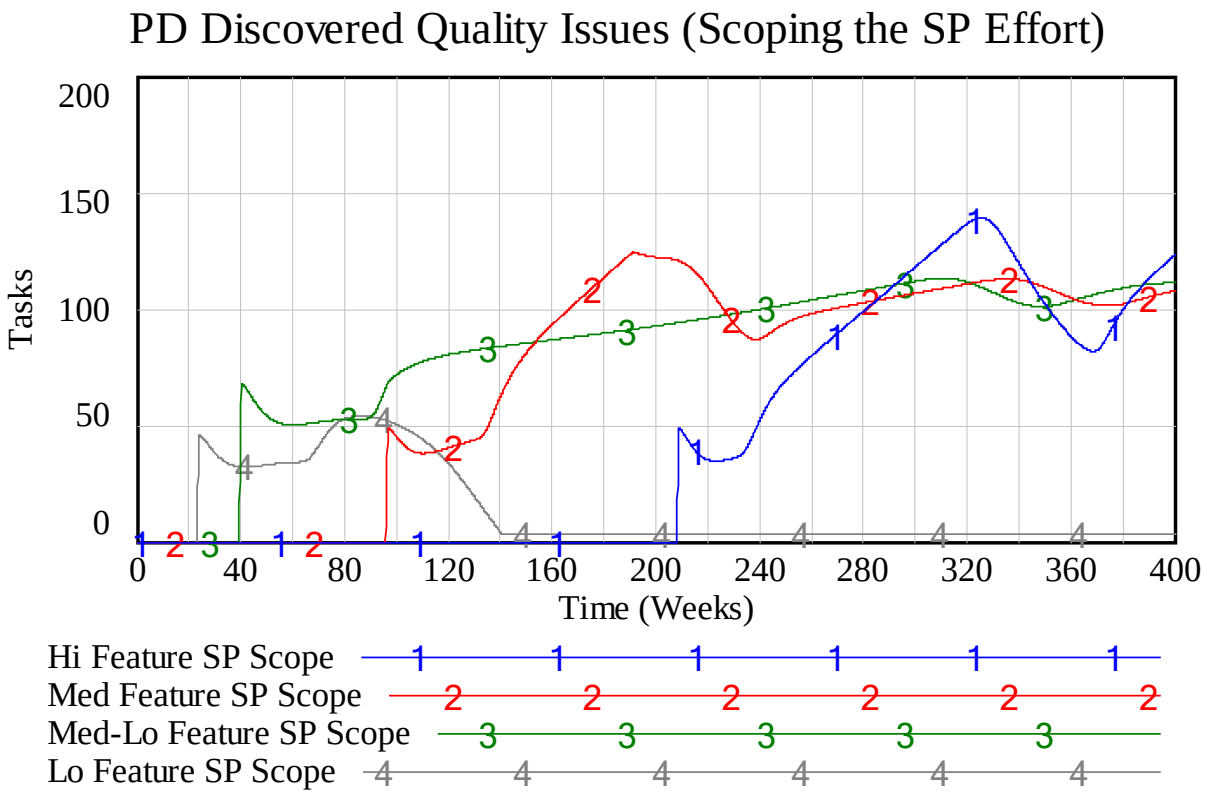
The tipping point contributes to the “90% Done” Syndrome



Key Preliminary Findings -4

The transition from science project to production effort should be made *early*

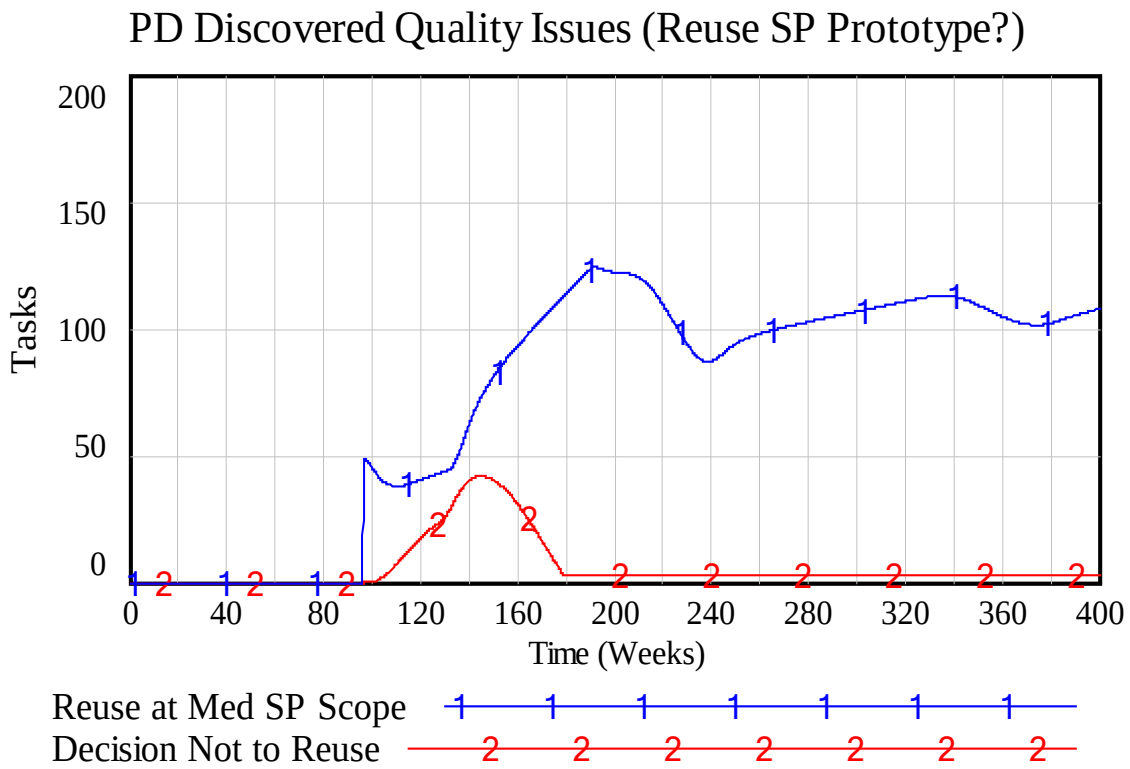
- A late transition increases the amount of undiscovered rework that is transferred



Key Preliminary Findings -5

Throwing away the prototype results in better program performance

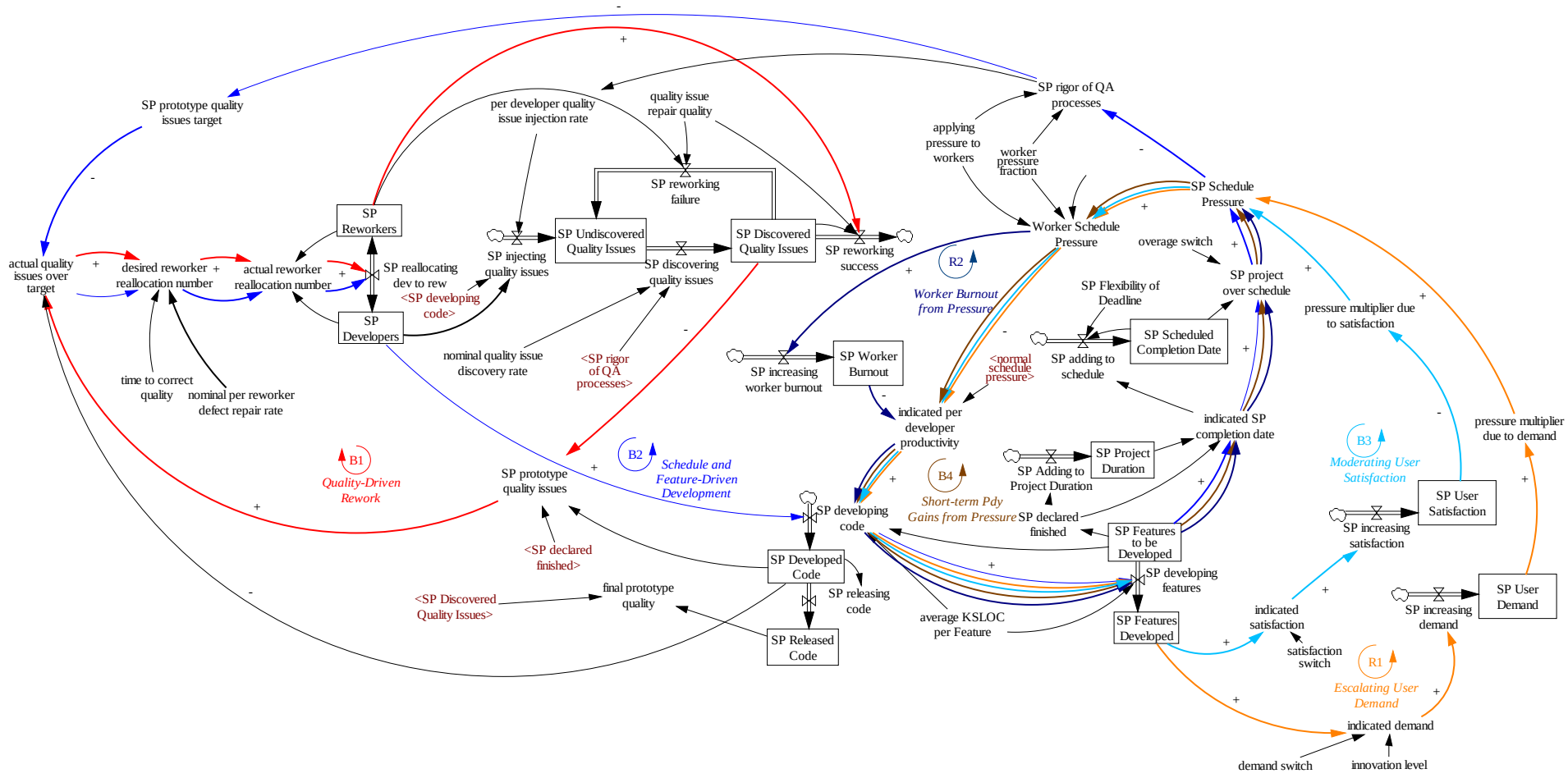
- However, very early transition or evolutionary development may also be viable



The Simulation Model

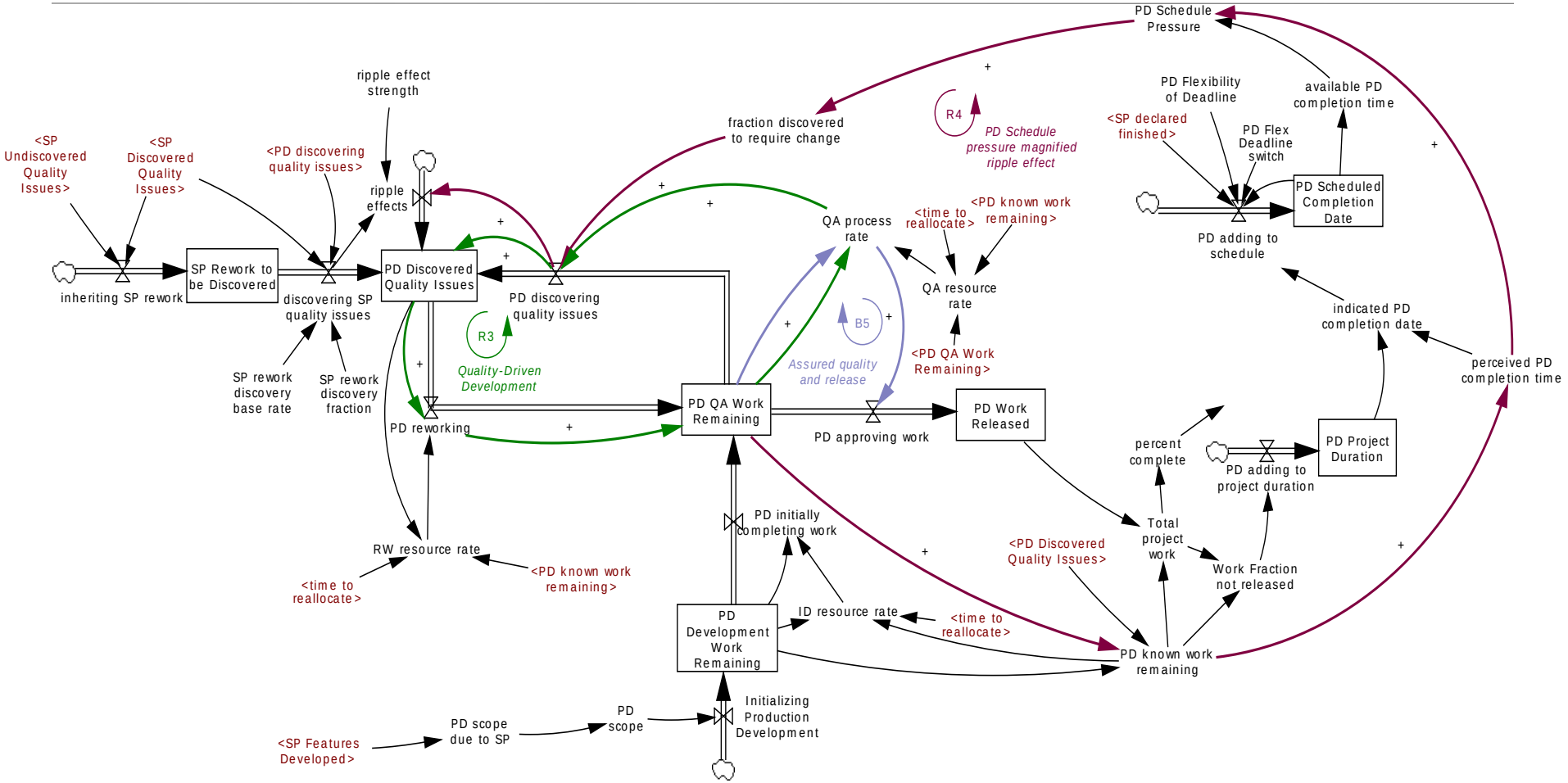
The Evolution of a Science Project

The Science Project (SP) Sector



The Evolution of a Science Project

The Production Development (PD) Sector *



* Adapted from the project management models produced by T. Taylor, D. Ford, S. Johnson, “Why Good Projects Go Bad: Managing Development Projects Near Tipping Points, ICSD 2005, Boston.

Preliminary Conclusions

Many acquisition program behaviors are an interaction of the inherent structural dynamics, inc. the incentives that act on key participants.

Tipping points may play a key role in many challenging software development and acquisition situations.

System dynamics modeling can provide a deep understanding of poorly understood dynamics in software-reliant acquisition.

Models are a great source for generating research questions!

Future work:

- Calibrating model based on acquisition program assessments conducted at SEI and data obtained from programs.
- Identifying and evaluating potential mitigations
- Develop games to teach model-based lessons learned

“Firefighting” Animation

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

Summary and Conclusions

Joint Program Acquisition Experience Wanted!

We are analyzing the dynamic organizational behavior of joint and joint-interest programs as part of an ongoing research project.

We are conducting group modeling workshops to elicit key joint program behaviors, and are using the information to build a system dynamics model.

If you'd be interested in participating in a workshop, or collaborating with us in other ways, please contact:

William E. Novak
Senior Member of Engineering Staff
Office: 412.268.5519
Email: wen@sei.cmu.edu
Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-3890

This material is based upon work supported by the U.S. Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Software Engineering Institute | Carnegie Mellon