

*The Opportunity Cost of
Solving Problems:
The Role of Testing in Product
Development*

Laura Black
International System Dynamics
Conference

Motivation

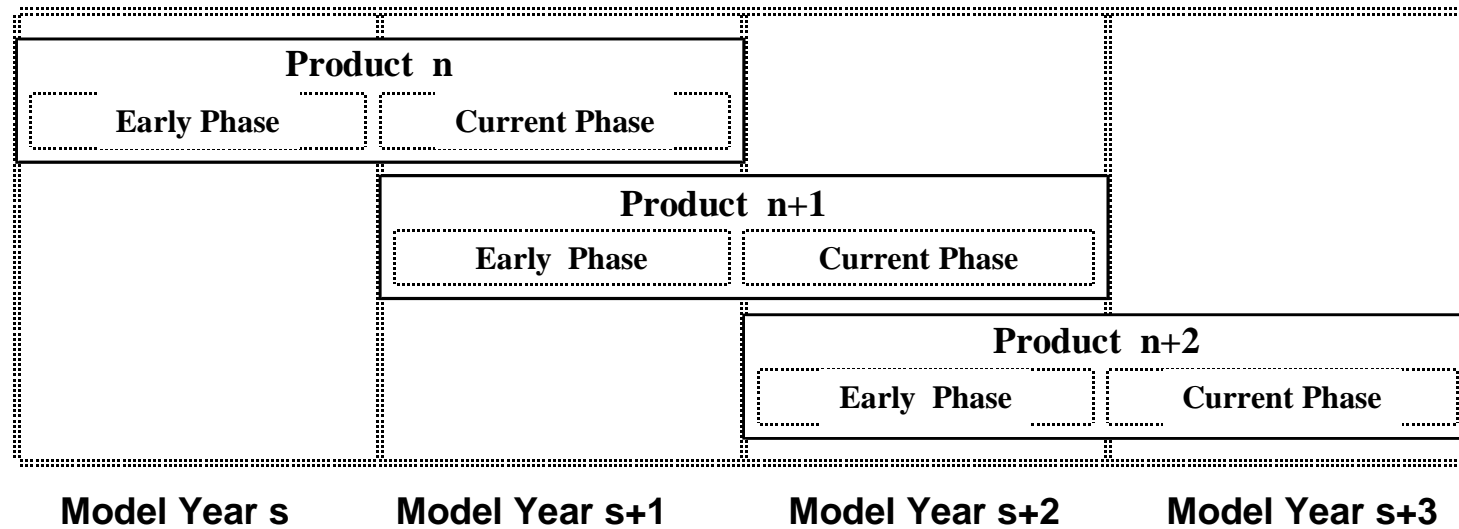
“We don’t know what we don’t know
until it’s *real* late.”

--Program Manager

Motivation

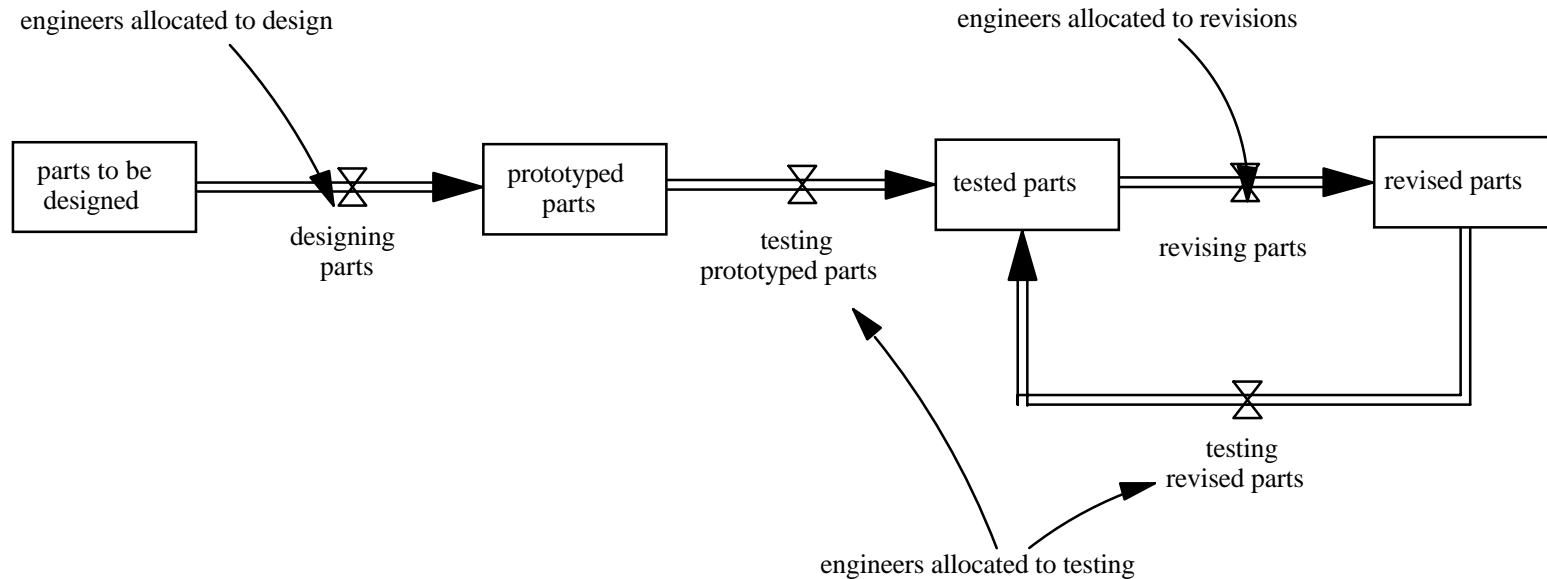
- Emergent dynamics in multiple projects difficult to manage
 - ✓ **Repenning's work:**
 - Scarce resources creates interdependence among projects
 - Transient increase in workload can lead to permanent decline in quality
- Example: Design and testing are iterative yet often treated as sequential.

Multi-project Environment

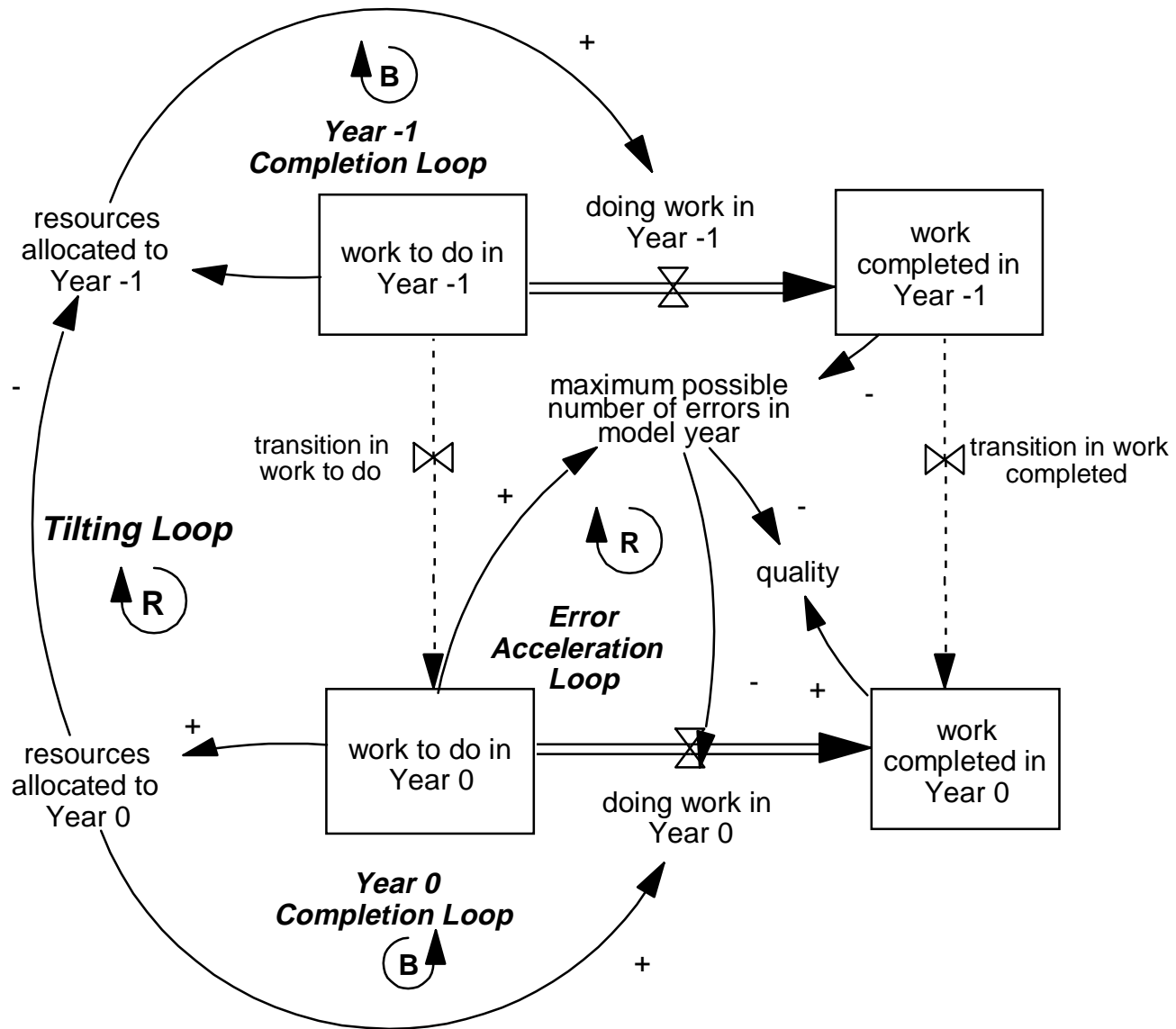


- Scarce resources for two projects at different phases
- Fixed annual launch date
- Bias toward work nearest deadline
- Early phase helps: lower probability of error

Modeling Assumptions

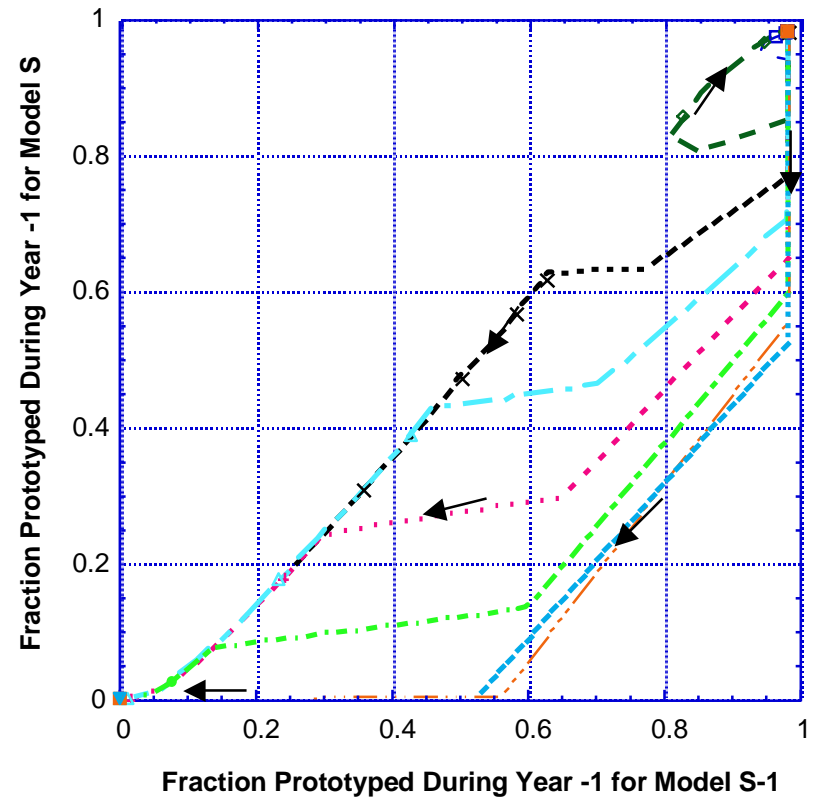
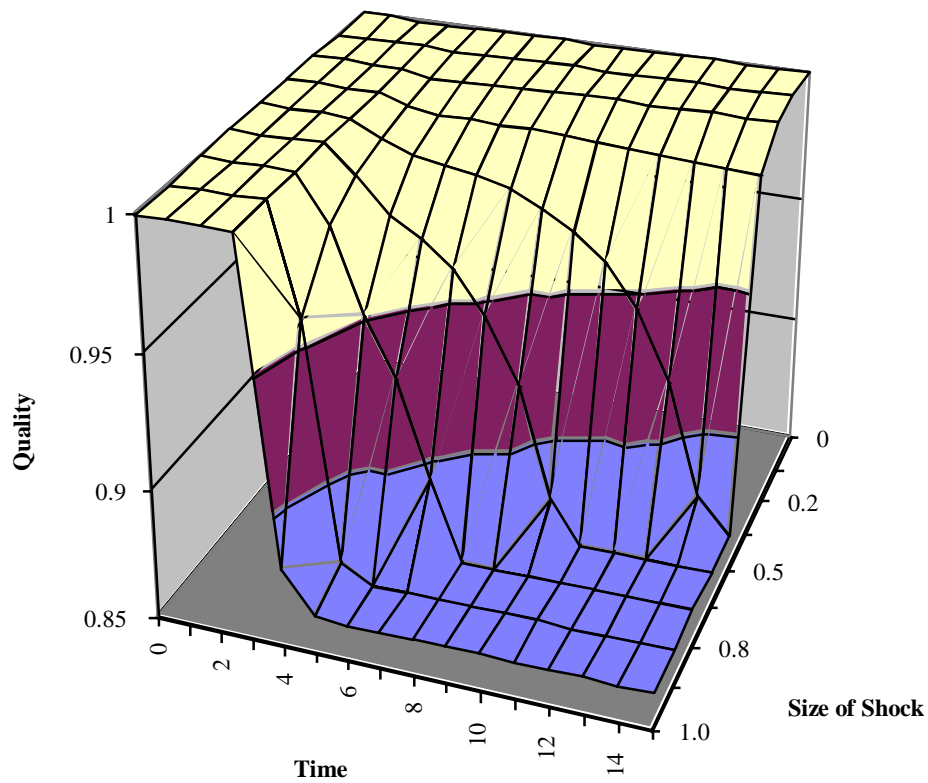


- Early and current phases structurally identical
- Initial priority for work
 - Current phase: design --> testing --> problem-solving
 - Early phase: design --> testing --> problem-solving



Base Case

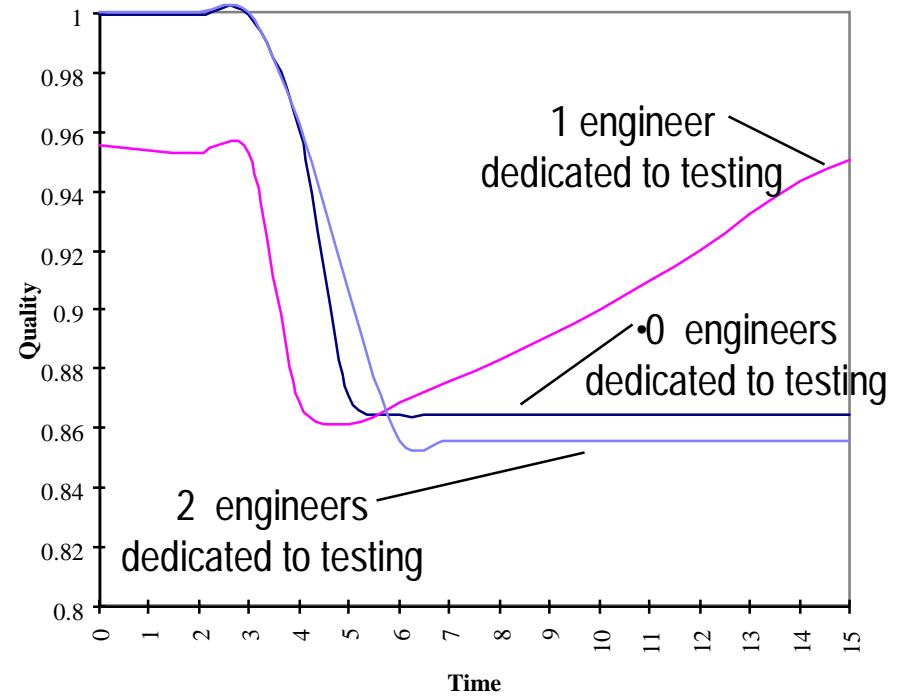
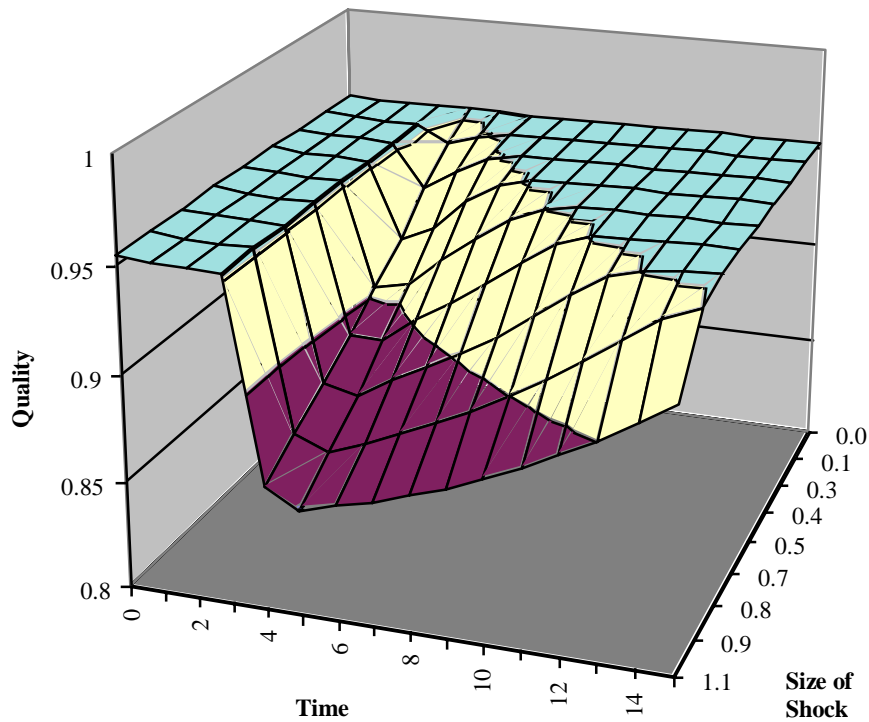
"Textbook" Allocation



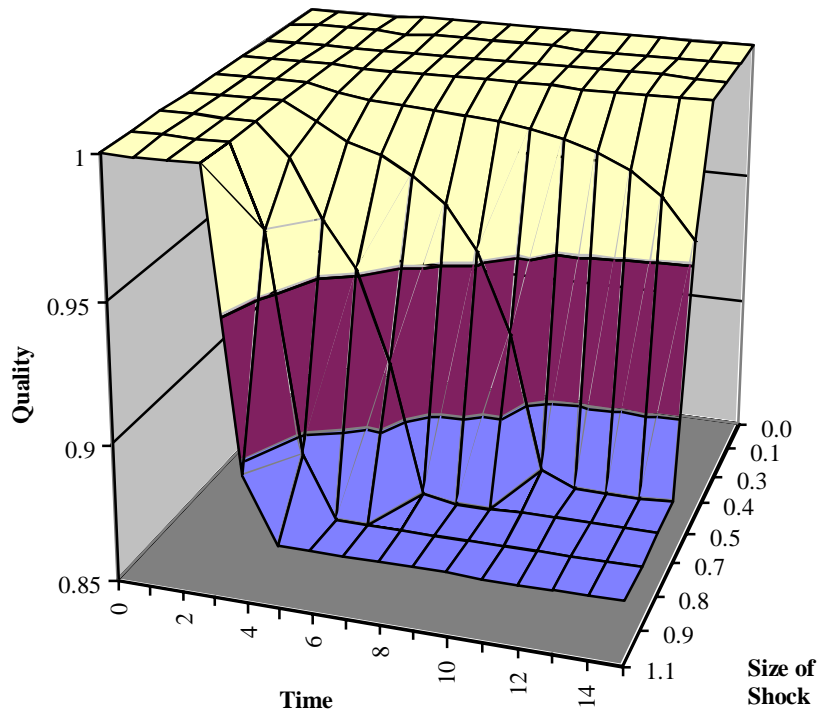
Policies Explored

- Dedicating resources to testing doesn't ensure high quality.
- Prototype build during last 12 months stabilizes system slightly.
- Altering priorities for resource allocation:
 - “Textbook” Allocation
 - ✓ Current design
 - ✓ Current testing
 - ✓ Current problem-solving
 - ✓ Early design
 - ✓ Early testing
 - ✓ Early problem-solving
 - “Urgent First” Allocation
 - ✓ Current problem-solving
 - ✓ Current design
 - ✓ Early problem-solving
 - ✓ Early design
 - ✓ Current testing
 - ✓ Early testing

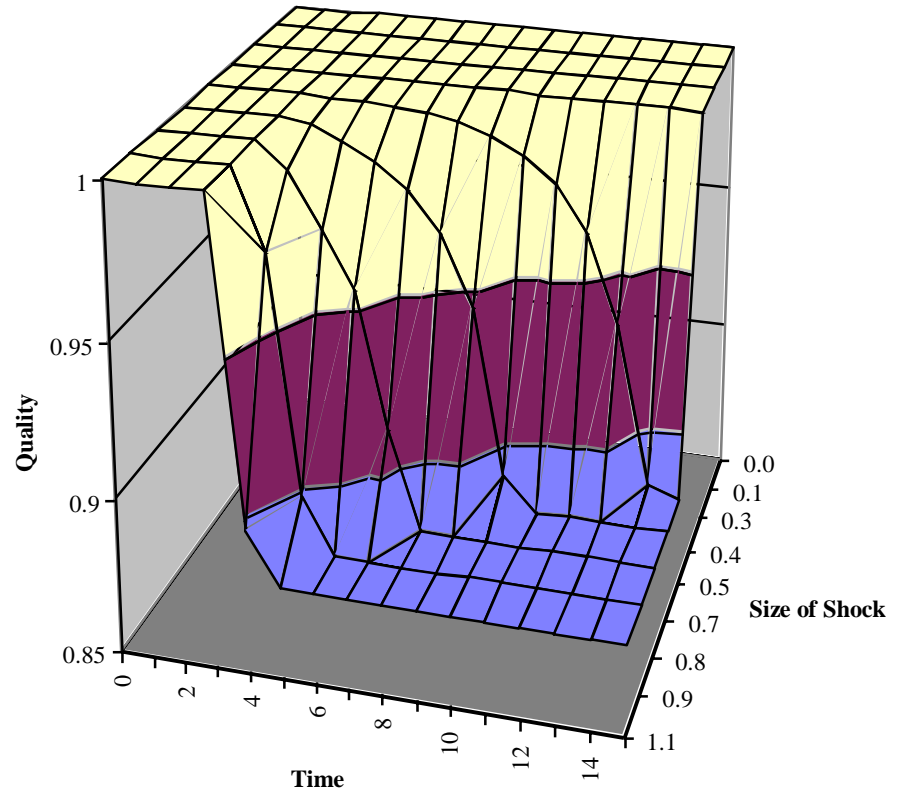
Dedicating People to Testing



With Prototype Build



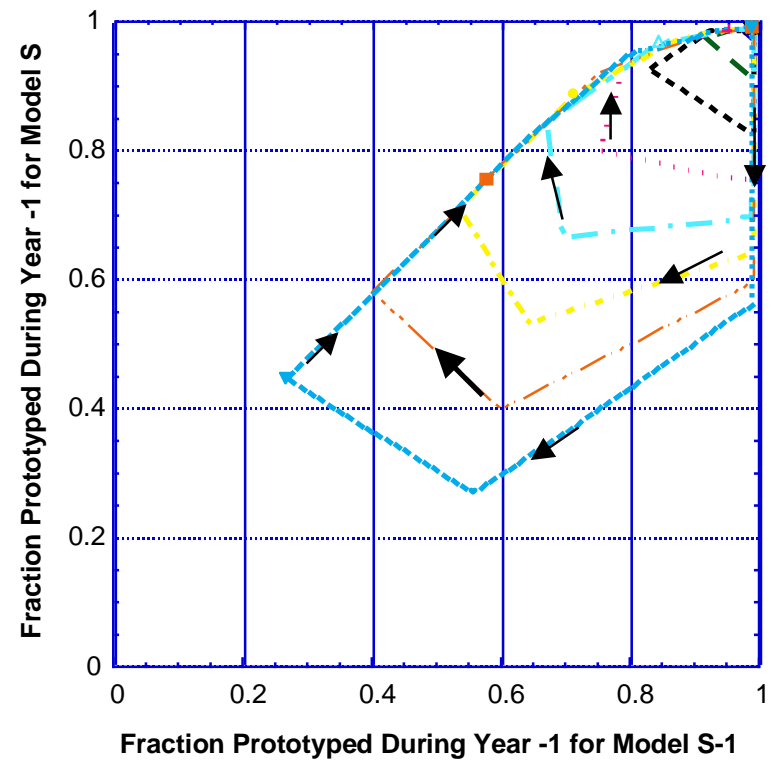
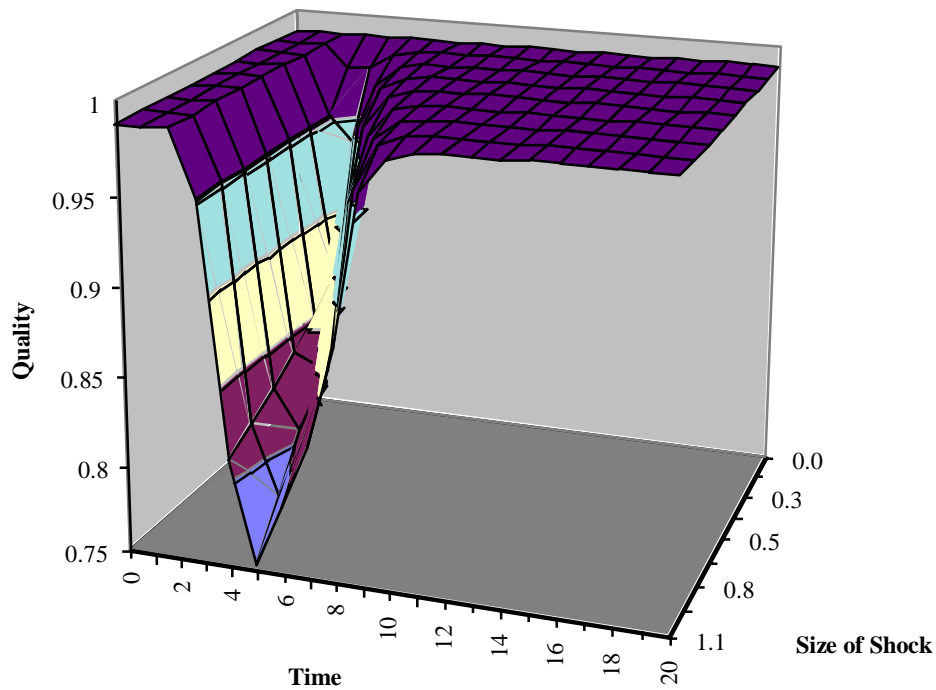
Build 34 Weeks Before Launch



Build 16 Weeks Before Launch

Varying Priorities

“Urgent First” Allocation

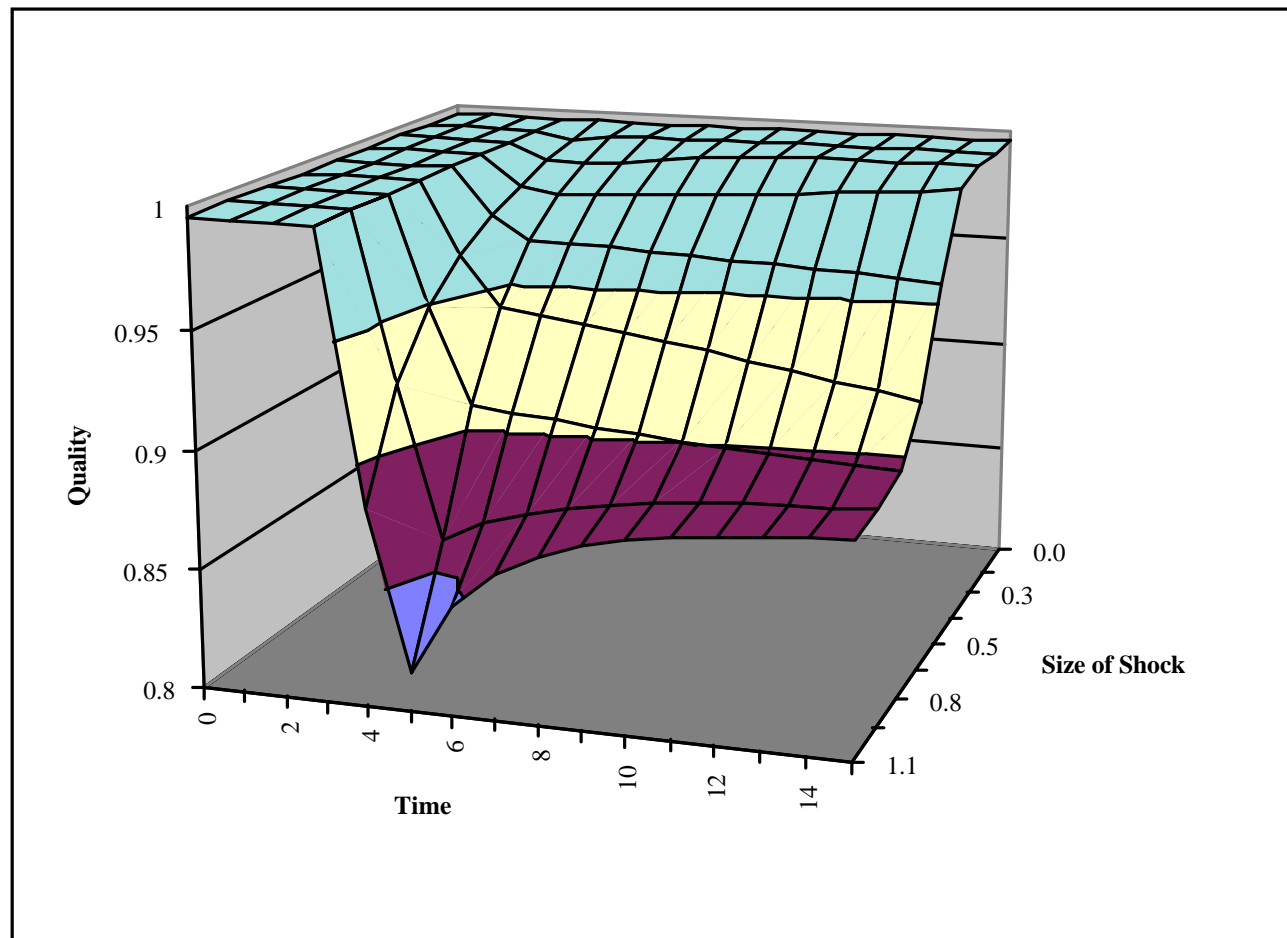


Policies Explored

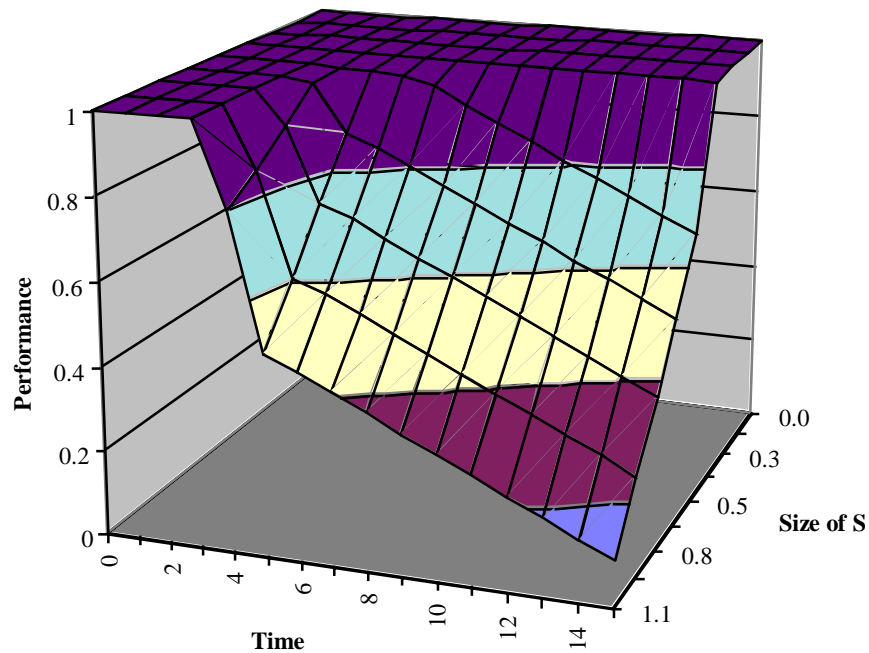
(continued)

- Fixing fraction of problems identified stabilizes system more, with a higher equilibrium after a dip in quality.
- Postponing incomplete work keeps quality high but at the expense of a reduced model year.
- Canceling incomplete work yields immediate performance hit but no long-term deterioration.

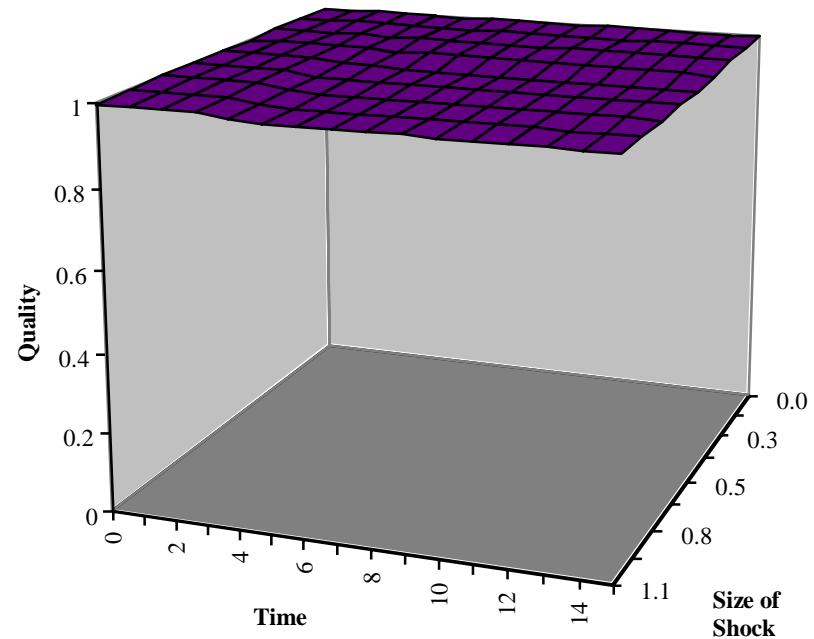
Fixing a Fraction of the Errors



Postponing Work

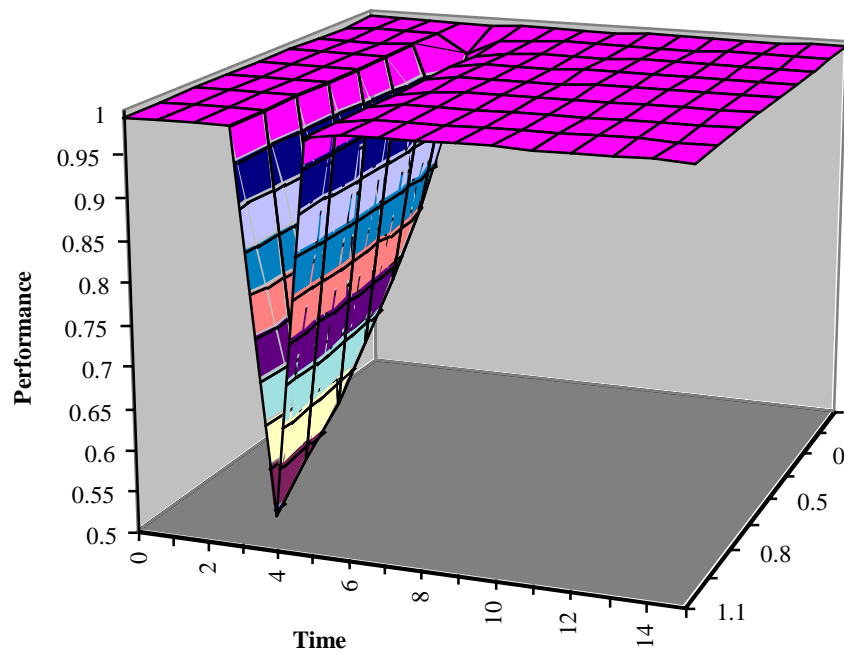


Performance
Decision 26 Weeks Before Launch

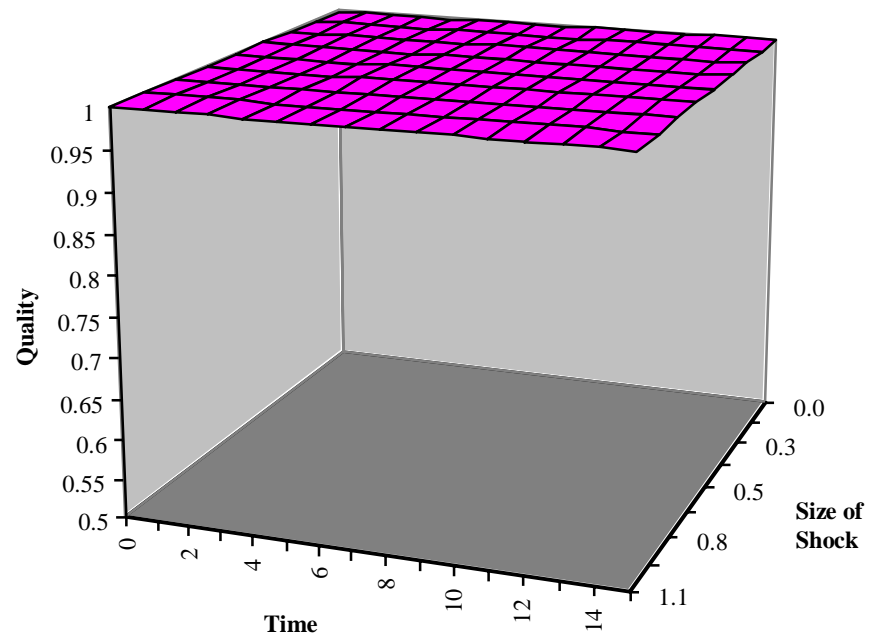


Quality
Decision 26 Weeks Before Launch

Canceling Work



*Performance
With Checkpoint 12 Months Before Launch*



*Quality
With Checkpoint 12 Months Before Launch*

Conclusions

- Balancing work-to-do with resources available assures long-term productive capacity.
 - ✓ **Managers can't balance what they're unaware of:**

**Parts to Be Designed + Identified Problems
= Total Work-to-Do**

- ***Additional cycles of testing and problem-solving can overload the development system as much as too many projects.***

Recommendations

- Dynamically allocating resources can help--if early work receives priority.
- Test plans and results can serve as cross-functional communication about total work-to-do.
- Targets for "what we don't know" raise awareness of the need to test early in the development cycle.
- Metrics for problem resolution can aid project management:
 - ✓ **Prioritizing problems identified**
 - ✓ **Average time to resolve problem**
 - ✓ **Percent of component-level concerns identified in build**