How much should we spend on quality assurance?

Stan Rifkin
MASTER SYSTEMS INC.
2604B El Camino Real 244
Carlsbad, California 92008 USA
+1 760 729 3388 sr @ Master-Systems.com

Same question as:

- How much quality assurance is enough?
- When should we stop testing?
- What is the relationship between product quality and the quality assurance process?

- Answer: It depends.

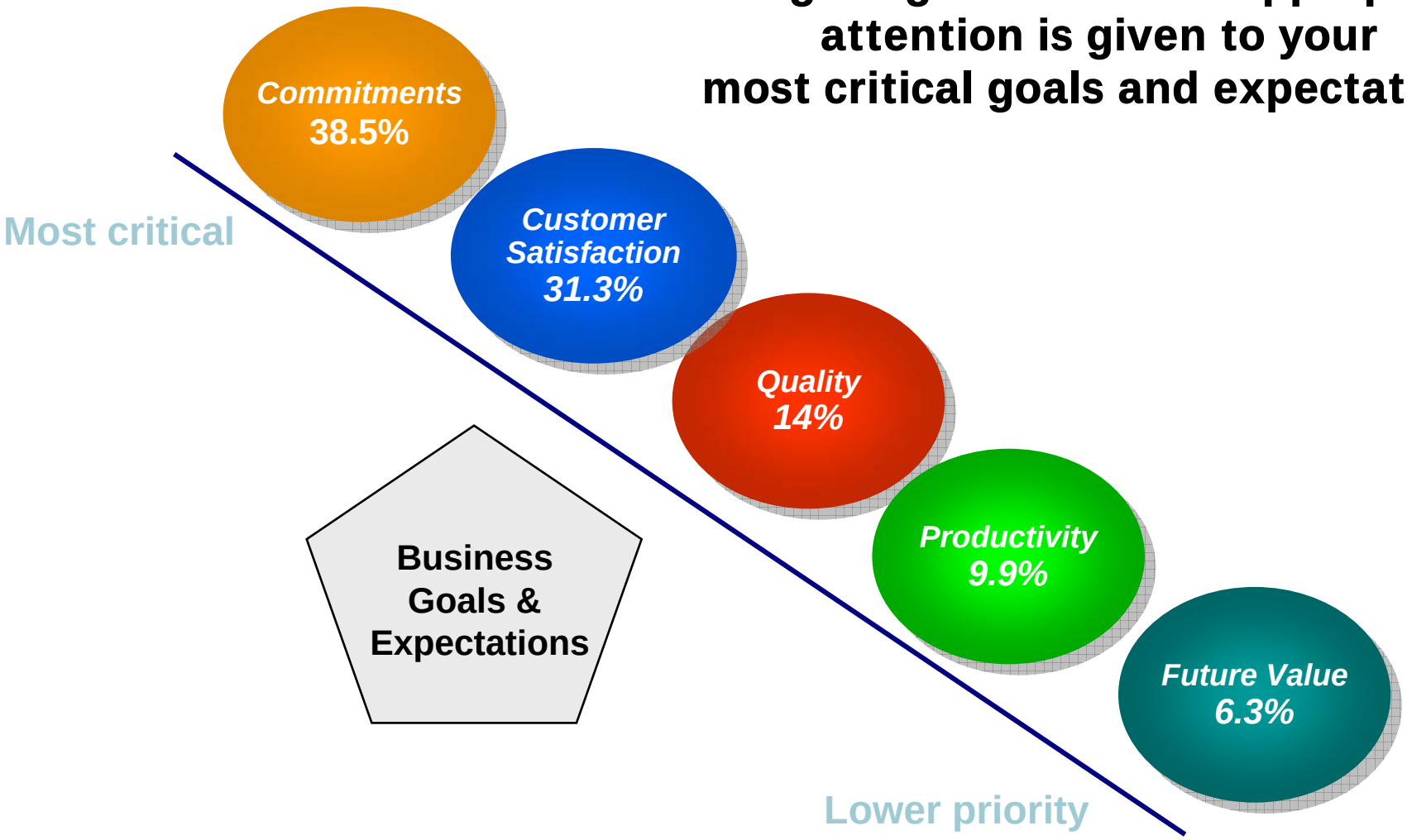
Going to speak about a new approach: value-based

- Remarks will be brief.
- Reports on the work of Barry Boehm and his PhD student LiGuo Huang, who graduates in a few months.
- Paper will be published in IEEE *Software* this year. Further results will be presented at the International Conference on Software Engineering in May.
- Codification of what some of us already know & do.
- A promising avenue of research, already with some concrete application.
- A way to think.

- The future!

CSC Balanced Scorecard Process

Weighting ensures that appropriate attention is given to your most critical goals and expectations



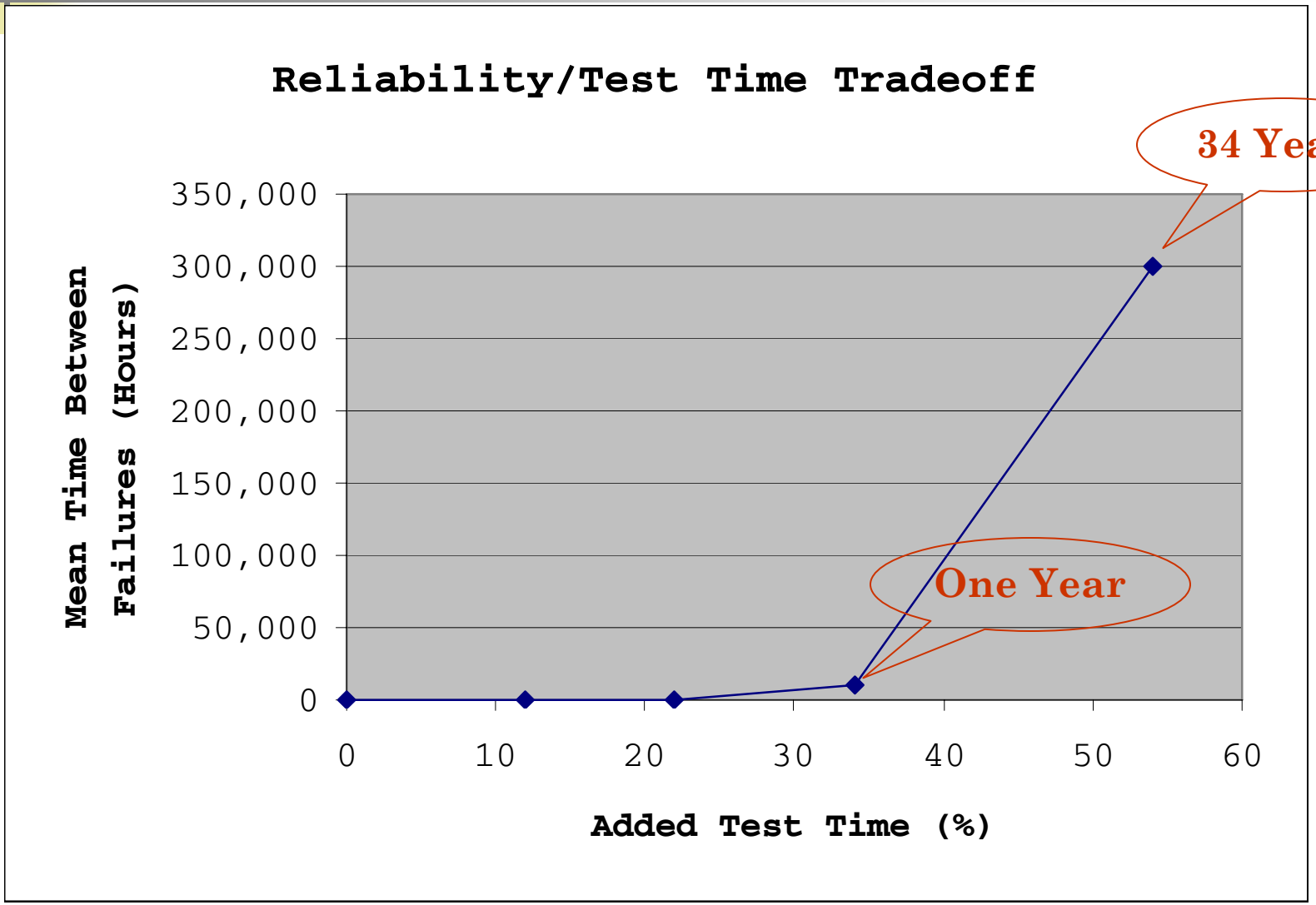
CSC Balanced Scorecard Process

- Brilliant process, based on a clever, seamless synthesis of many best practices.
- BUT, what do I do every day to achieve the results?
- What actions should I take in order to achieve the goals?

Enter: Value-based software engineering

- The problems it is trying to solve:
  - Canceled projects – after large investment.
  - Inefficient projects (e.g., Death March)
- Limitations:
  - Method independent.
  - Cannot solve all problems.
  - More notional than detailed today, in general.
- Solution approach
  - Step-by-step directions for selecting important aspects of the product, process, technology, and human resources.
  - Step-by-step guidance on what to do to achieve win-win outcome.

Example: Value of added testing



Source: COCOMO II values for RELY, the reliability required of the software product.

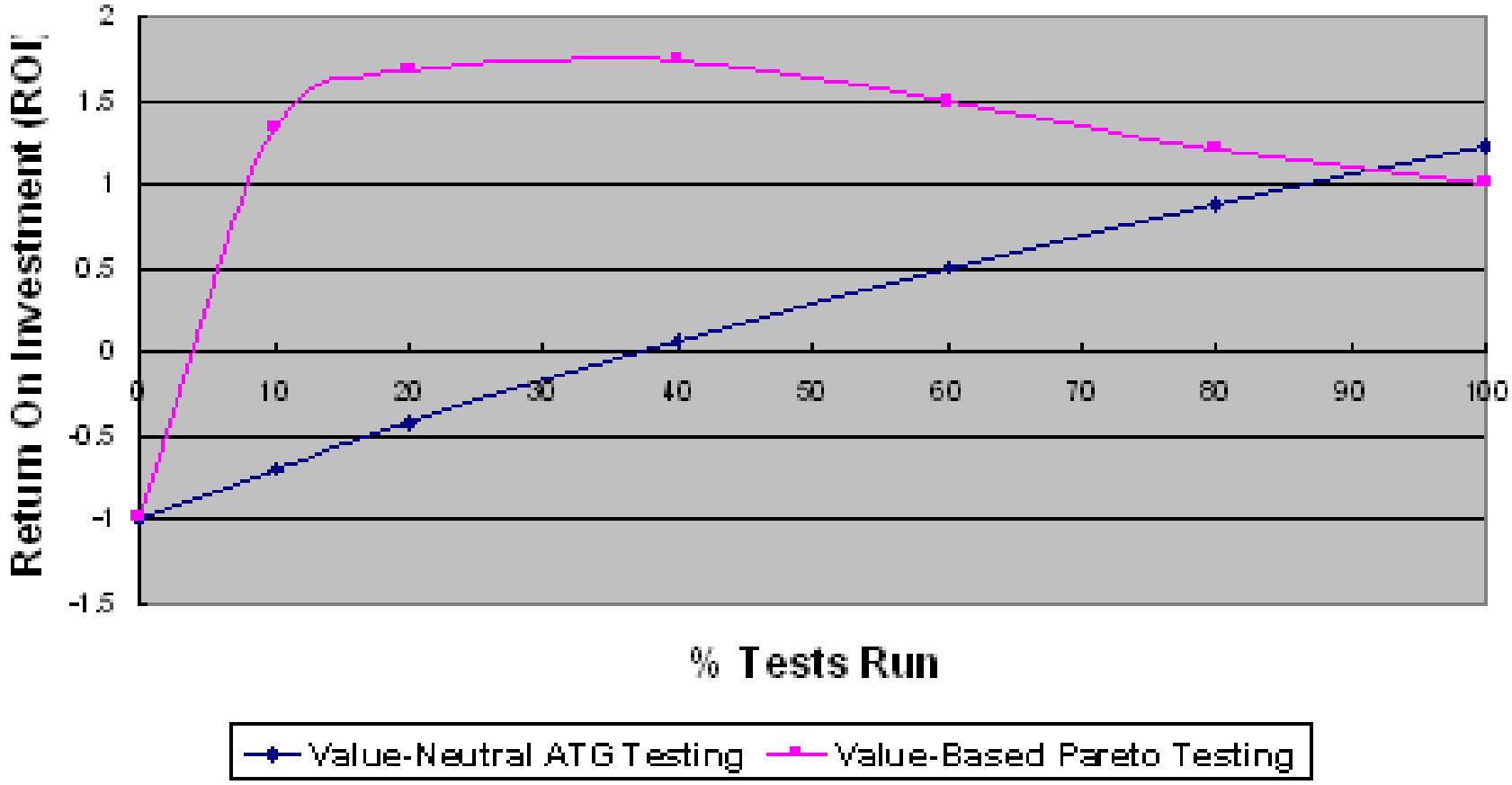
What would you do with the additional test time?

Table 1. Defect Removal Investment Rating Scales

Rating	Automated Analysis	Peer Reviews	Execution Testing and Tools
Very Low	Simple compiler syntax checking.	No peer review.	No testing.
Low	Basic compiler capabilities	Ad-hoc informal walkthroughs	Ad-hoc testing and debugging.
Nominal	Compiler extension Basic requirements and design consistency	Well-defined sequence of preparation, review, minimal follow-up.	Basic test, test data management, problem tracking support. Test criteria based on checklists.
High	Intermediate-level module and inter-module; Simple requirements/design	Formal review roles with well-trained participants and using basic checklists, follow up.	Well-defined test sequence tailored to organization. Basic test coverage tools, test support system. Basic test process management.
Very High	More elaborate requirements/design Basic distributed-processing and temporal analysis, model checking, symbolic execution.	Basic review checklists, root cause analysis. Formal follow-up using historical data on inspection rate, preparation rate, fault density.	More advanced test tools, test data preparation, basic test oracle support, distributed monitoring and analysis, assertion checking. Metrics-based test process management.
Extra High	Formalized specification and verification. Advanced distributed processing	Formal review roles and procedures. Extensive review checklists, root cause analysis. Continuous review process improvement. Statistical Process Control.	Highly advanced tools for test oracles, distributed monitoring and analysis, assertion checking Integration of automated analysis and test tools. Model-based test process management.

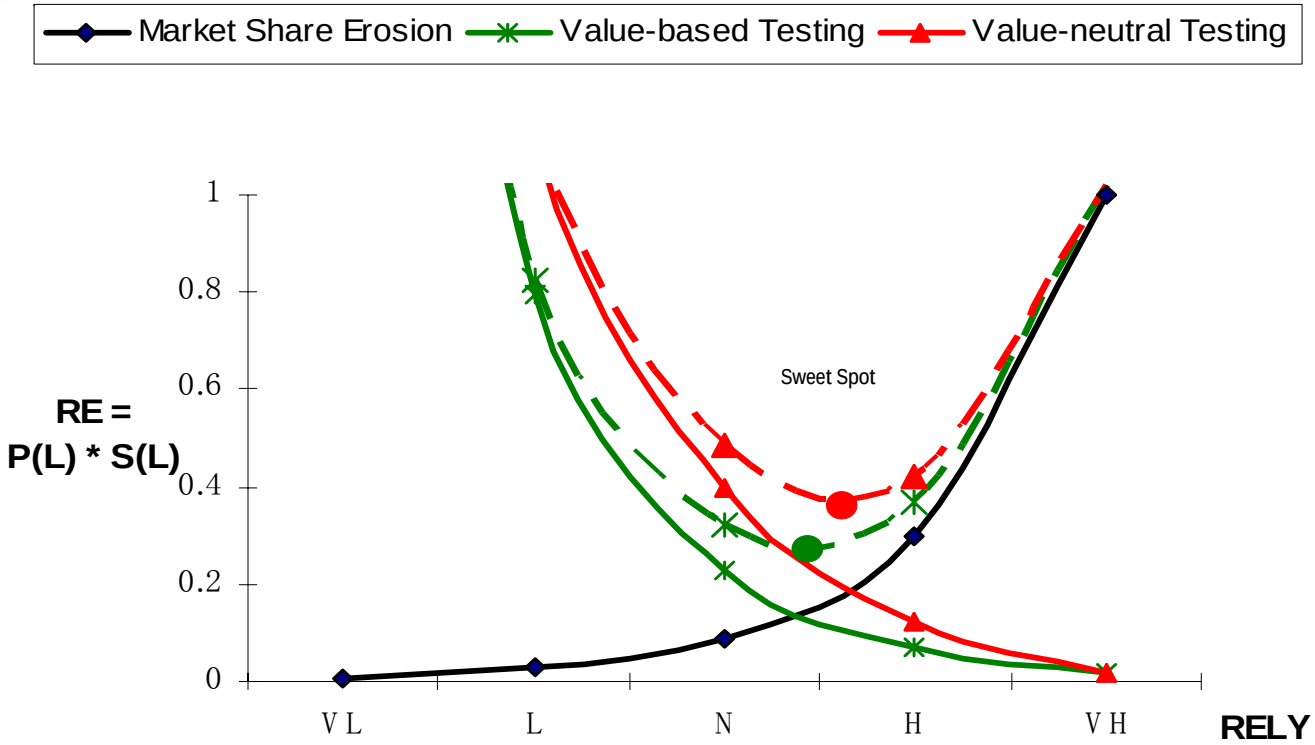
Source: How Much Software Assurance is Enough: A Value-Based Approach, LiGuo Huang & Barry Boehm, IEEE *Software*, 2006, to appear.

ROI on VBSE testing: There is an optimum, given the goal



Source: Huang & Barry Boehm, op. cit.

Comparing Value-Based Testing vs. Value-Neutral Testing

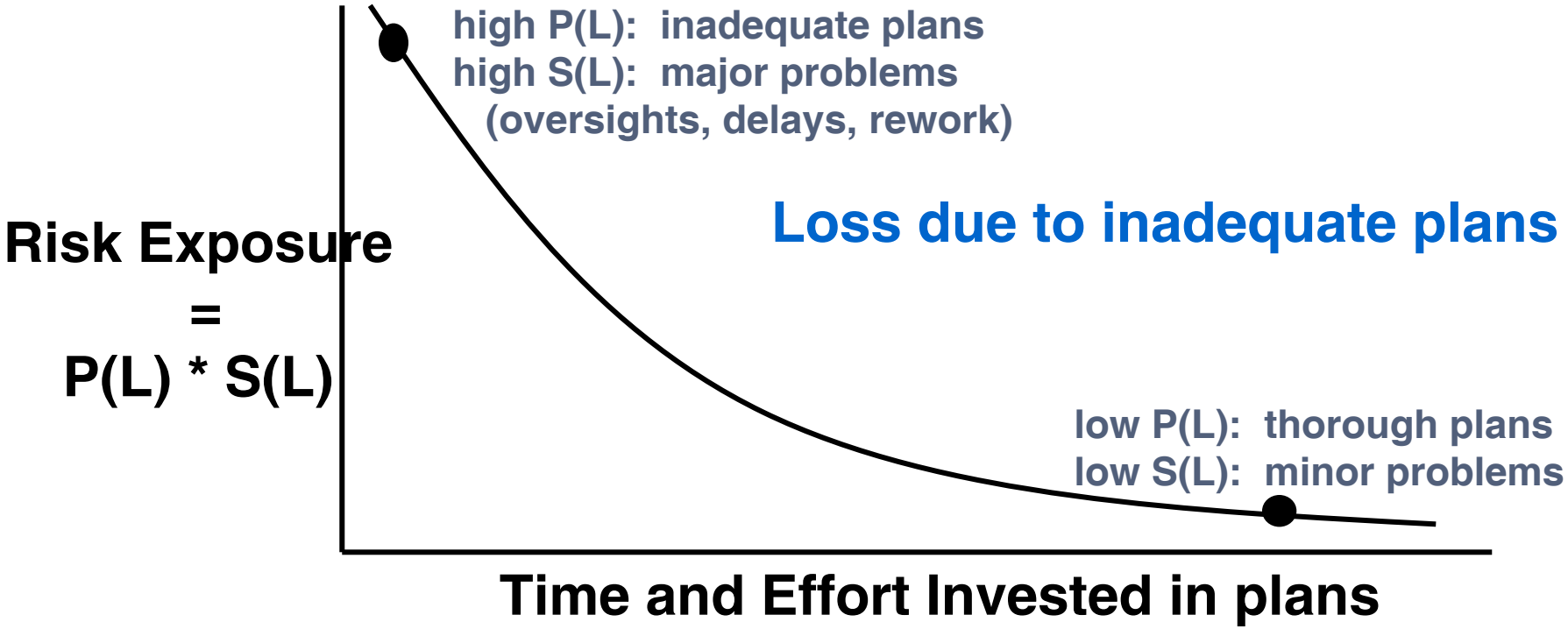


COCOMO II:	**0**	12	22	34	54	**Added % test time**
COQUALMO:	**1.0**	.475	.24	.125	.06	**$P_a(L)$**
Value-Based:	**3.0**	1.68	.96	.54	.30	**$S_a(L)$: Pareto**
Value-Neutral:	**3.0**	2.33	1.65	0.975	.30	**$S_a(L)$: Linear**
Market Risk:	**.008**	.027	.09	.30	1.0	**RE_m**

Source: Huang & Boehm, op. cit.

Example RE Profile: Planning Detail

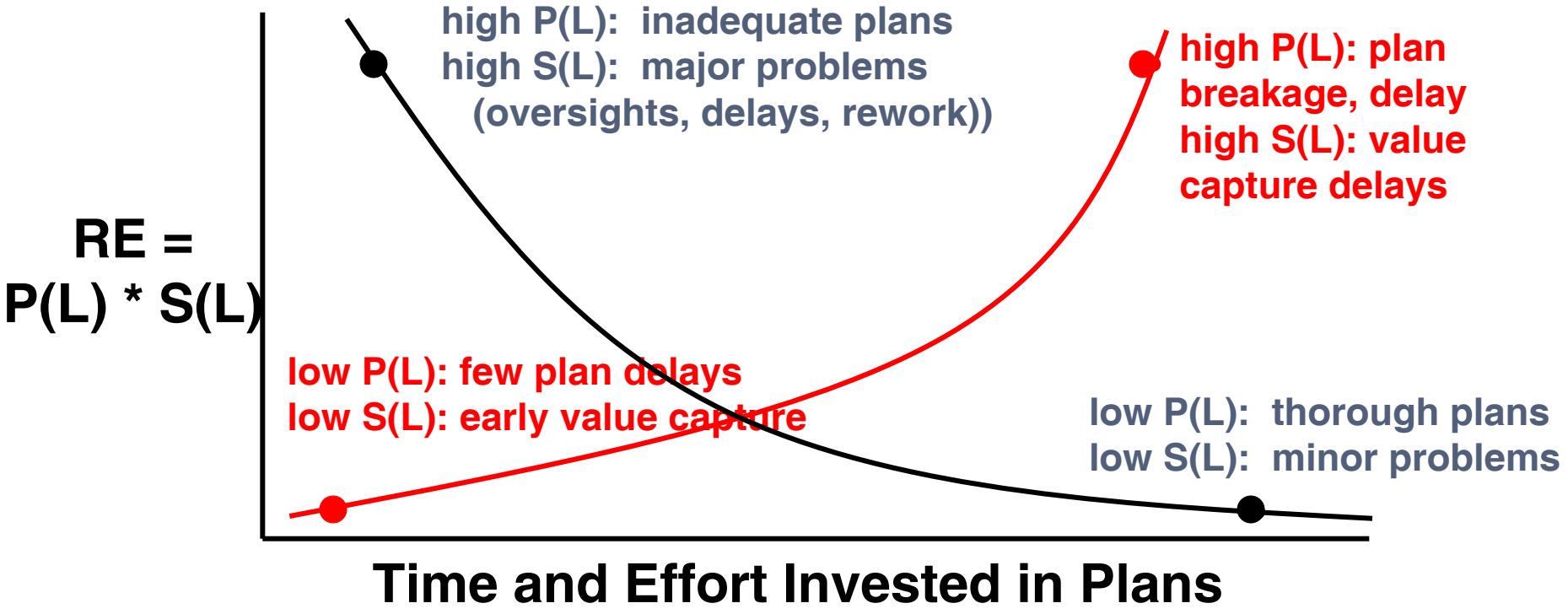
Risk exposure = Sum over all events of [Probability of event x size (impact) of event]



Source for this slide and the following four: Many of Barry Boehm's presentations and last year's SPIN presentation by Stan Rifkin, "What is the best way to develop software? Continuing the conversation about agility and plan-driven methods," June 2005.

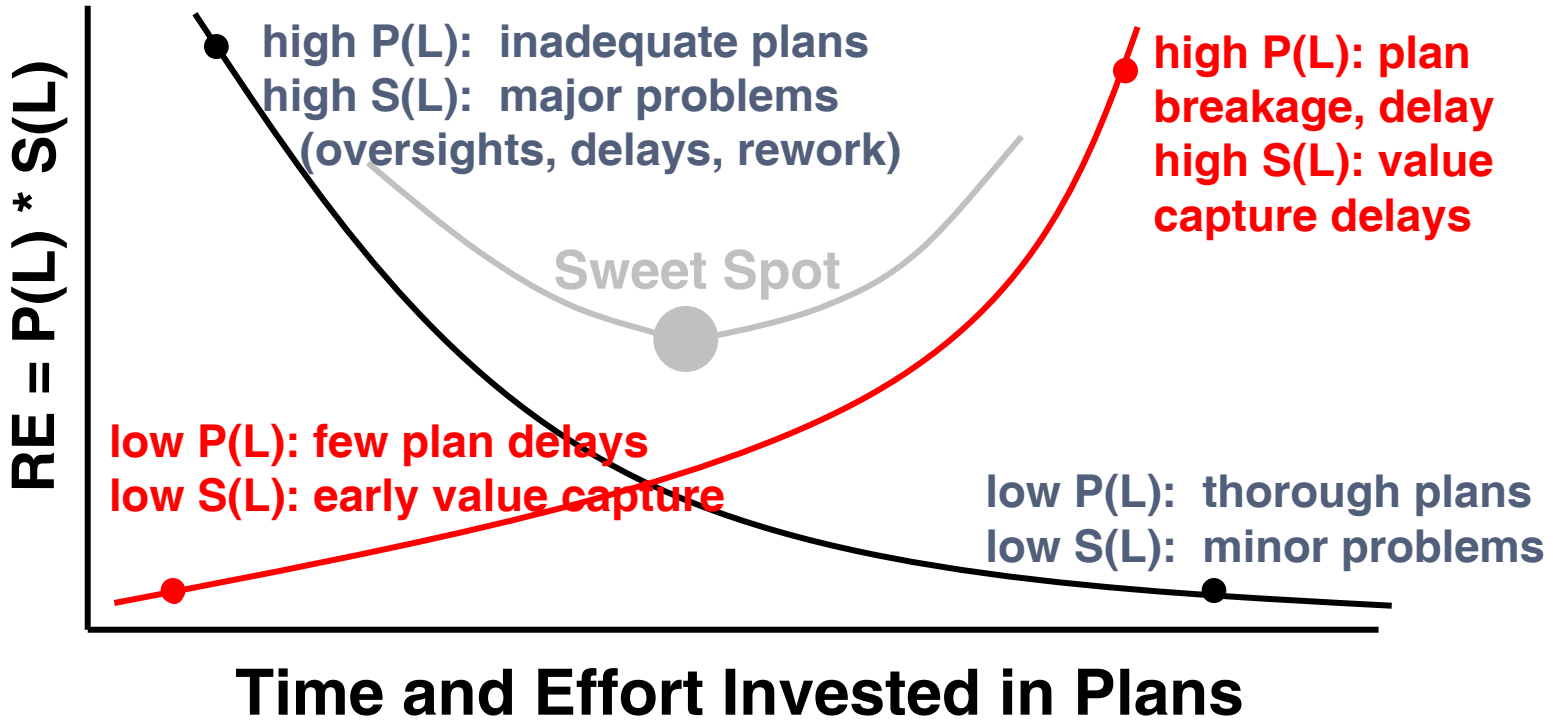
Example (cont.)

- Loss due to inadequate plans
- Loss due to market share erosion

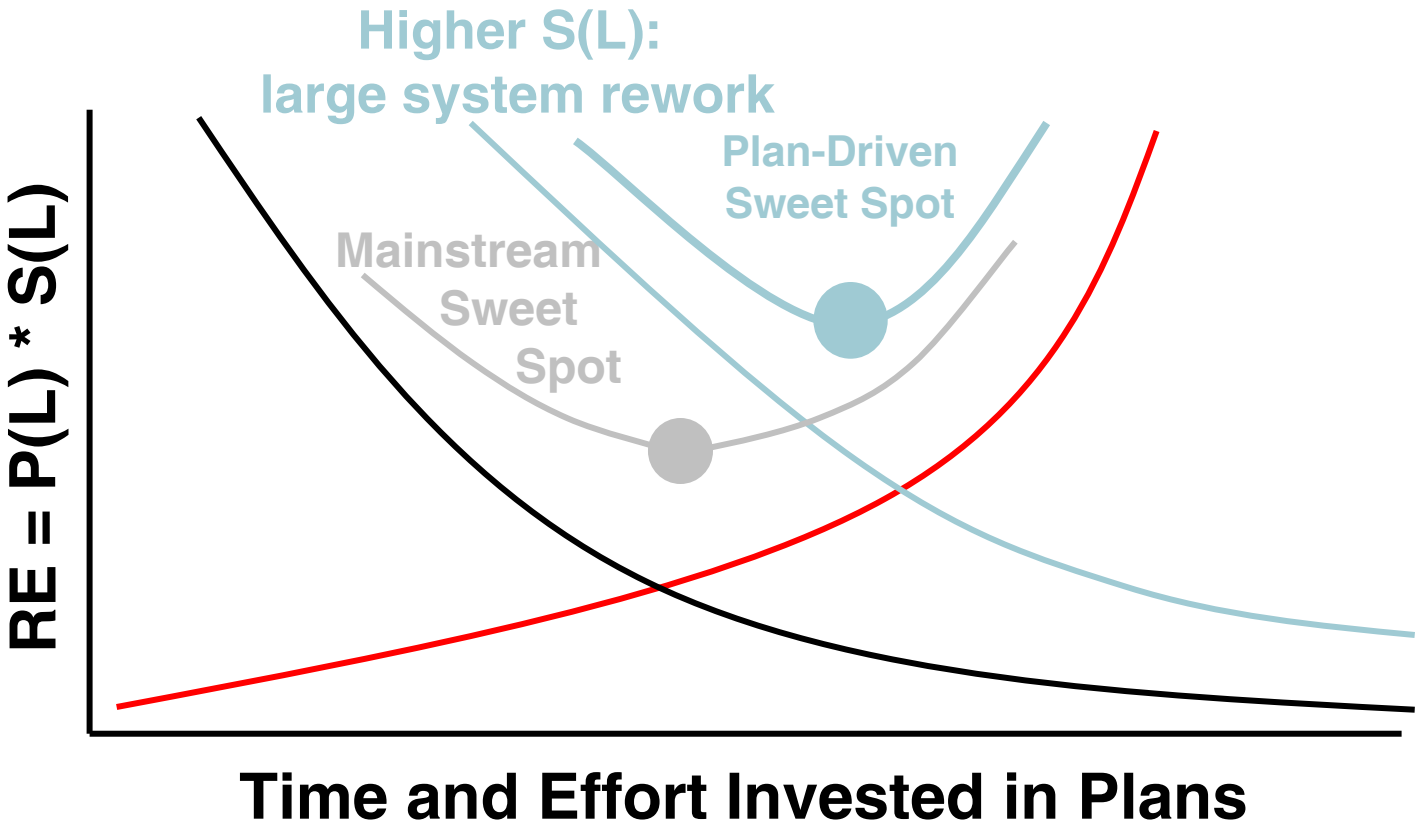


Example RE Profile: When to Ship

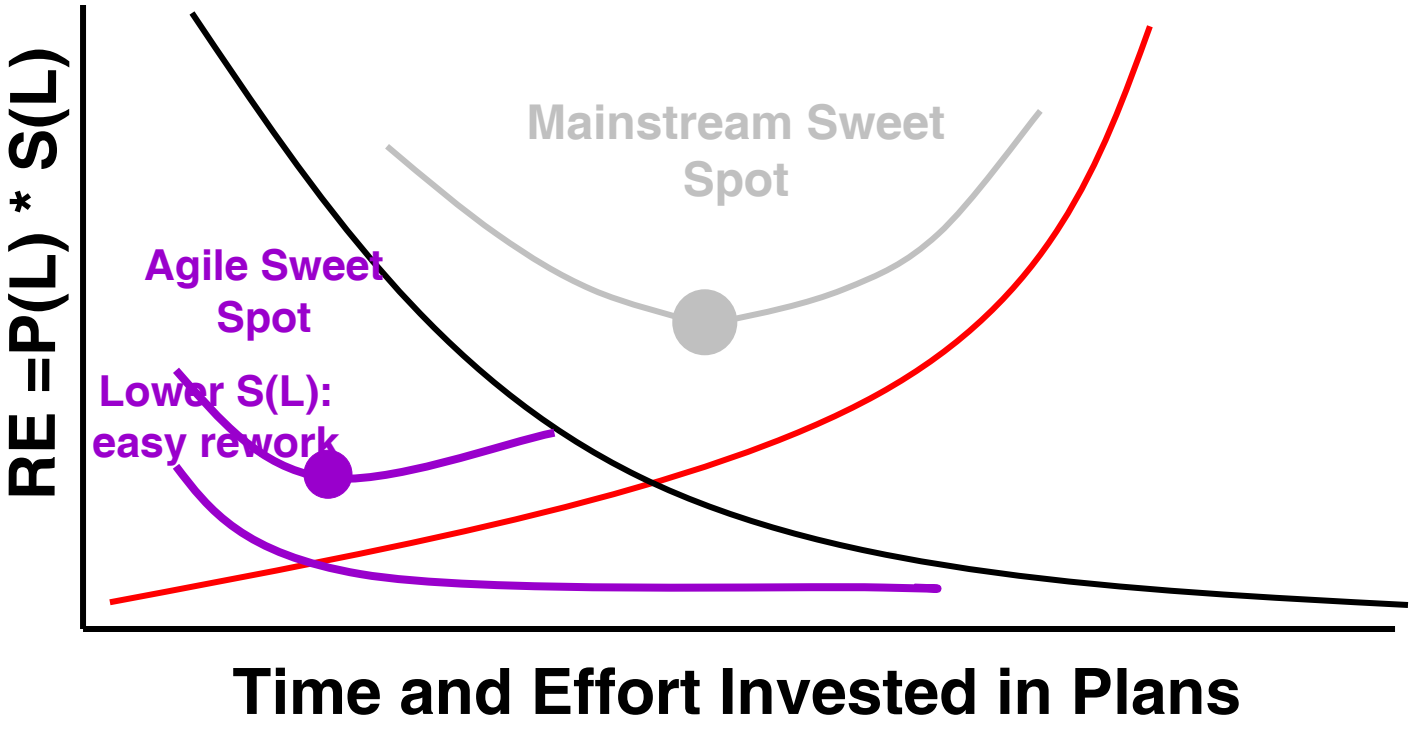
- Sum of Risk Exposures



Plan-Driven Home Ground



Agile Home Ground



Another example: Stakeholder synchronization vs. heads-down work

- If I synchronize often with stakeholders it is costly and I avoid rework.
- If I work with my head down I accomplish a lot, don't have to give "presentations," and I might be off-track for quite awhile.

- Is there an optimum mix?

Assume this life cycle

Process Milestones	Software Development Activities
Initiate project	Acquire system requirements
SCS define acceptable & desired values for Q-attributes	Requirement elicitation meeting
	Win-Win negotiation
Risk analysis & architecture/technology evaluation	Internal prototype evaluation
	External prototype evaluation
Identify conflicting Q-attributes & perform tradeoff analysis	
SCS adjust acceptable values for Q-attributes	Stakeholder renegotiation
System top-level design and initial Feasibility Rationale Description (FRD)	System top-level design
LCO Review	Architecture options internal review
	Architecture options external review
SCS refine acceptable & desired values for Q-attributes	Requirement elicitation meeting
	Win-Win negotiation
System detailed design and detailed Feasibility Rationale Description (FRD)	System detailed design & FRD
LCA Review	Selected architecture internal review
	Selected architecture external review
Core capability implementation	Core capability implementation
Value-based core capability testing	Internal core capability testing
CCD	Internal core capability demo
	On-site core capability demo
Remaining features implementation	Complete system implementation
IOC Acceptance Review	On-site System Acceptance Review

Legend:

Life Cycle Objective (LCO)

Life Cycle Architecture (LCA)

Core Capability Demo (CCD)

Initial Operational Capability (IOC)

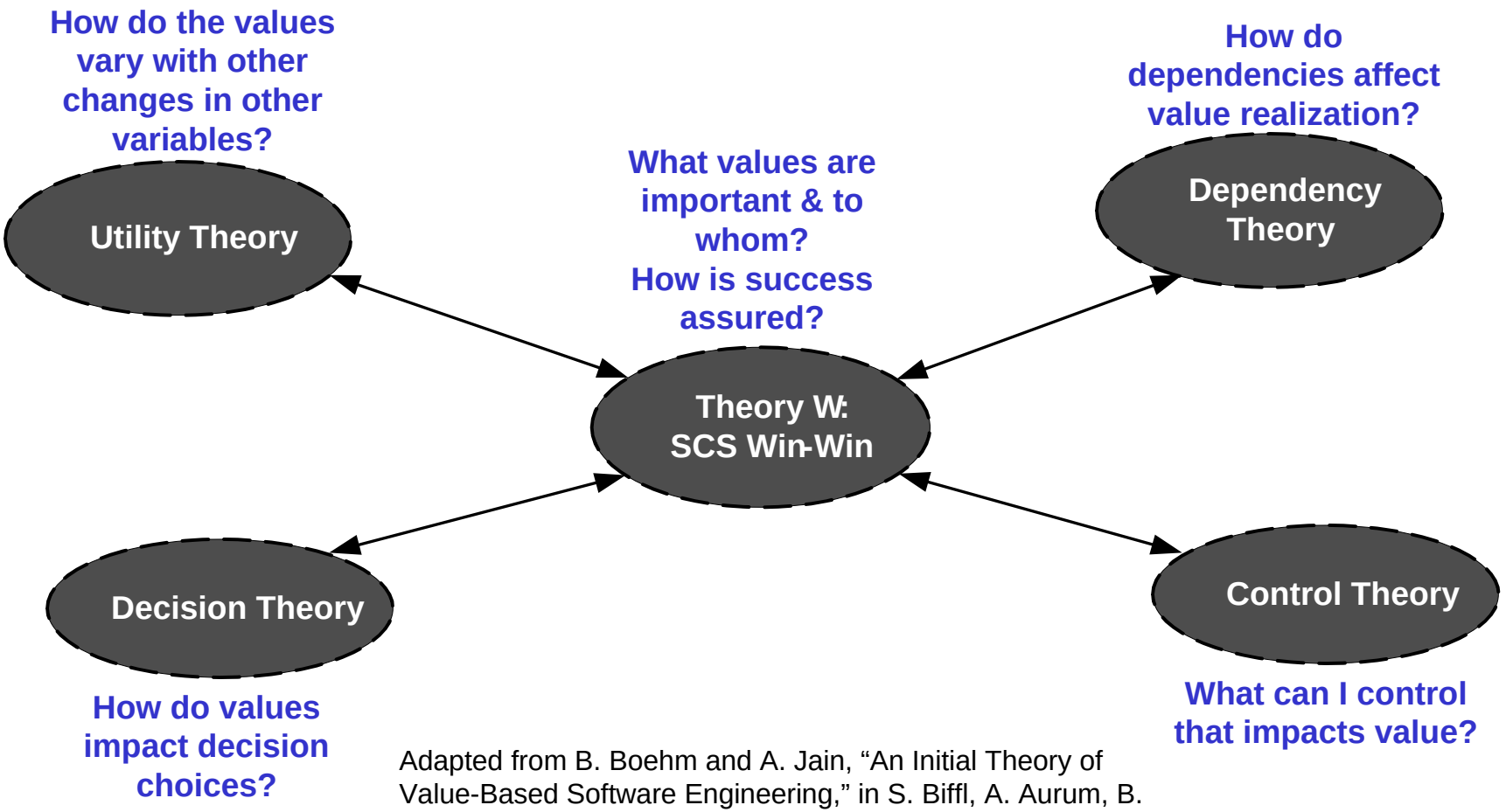
Source: Applying the Value/Petri Process to ERP Software Development in China, LiGuo Huang et al., ICSE 2006.

ROI on internal vs. external life cycle activities

	Process Activity Combinations	ROI
1	LCO(i)\ LCA(i) \ CCD(i) \ IOC(s)	—
2	LCO(s)\ LCA(i) \ CCD(i) \ IOC(s)	**6.2**
3	LCO(i)\ LCA(s) \ CCD(i) \ IOC(s)	2.4
4	LCO(i)\ LCA(i) \ CCD(s) \ IOC(s)	0.1
5	LCO(s)\ LCA(s) \ CCD(i) \ IOC(s)	**6.2**
6	LCO(s)\ LCA(i) \ CCD(s) \ IOC(s)	**5.8**
7	LCO(i)\ LCA(s) \ CCD(s) \ IOC(s)	2.3
8	LCO(s)\ LCA(s) \ CCD(s) \ IOC(s)	5.5

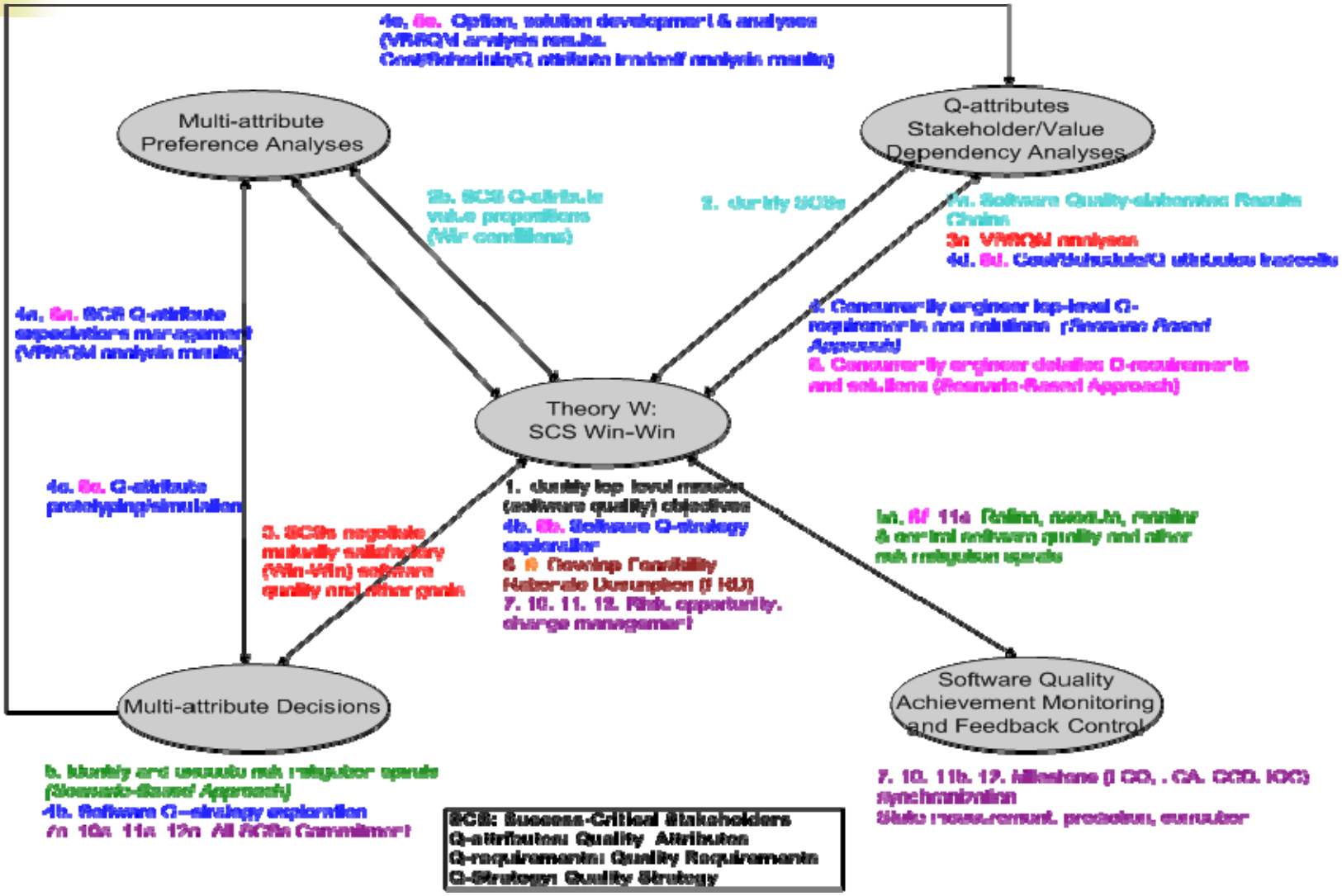
Source: Applying the Value/Petri Process to ERP Software Development in China, LiGuo Huang et al., ICSE 2006.

4 + 1 Framework



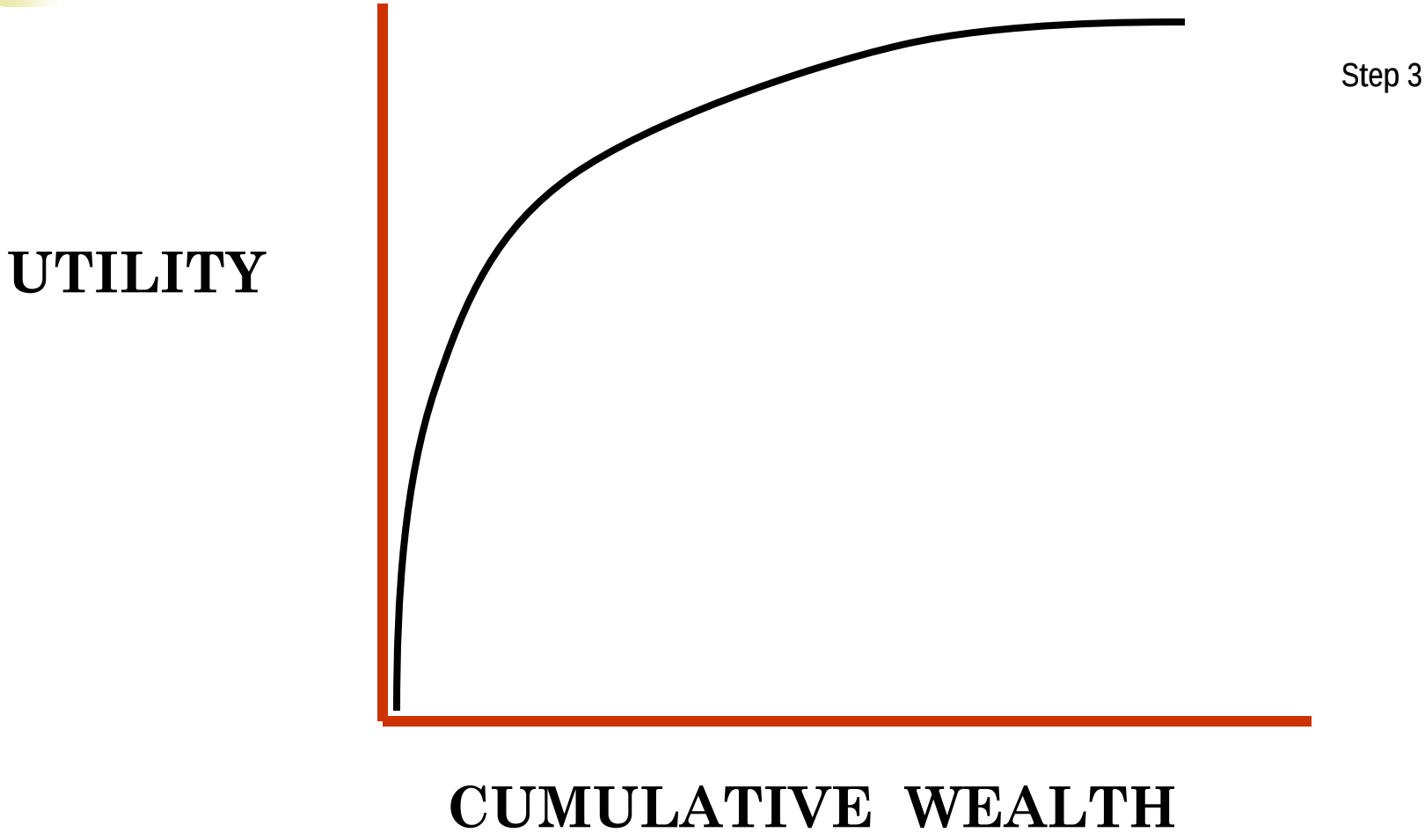
Adapted from B. Boehm and A. Jain, "An Initial Theory of Value-Based Software Engineering," in S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Gruenbacher (eds), Value-Based Software Engineering, Springer, 2005, pp. 15-37.

7 Step process of VBSE



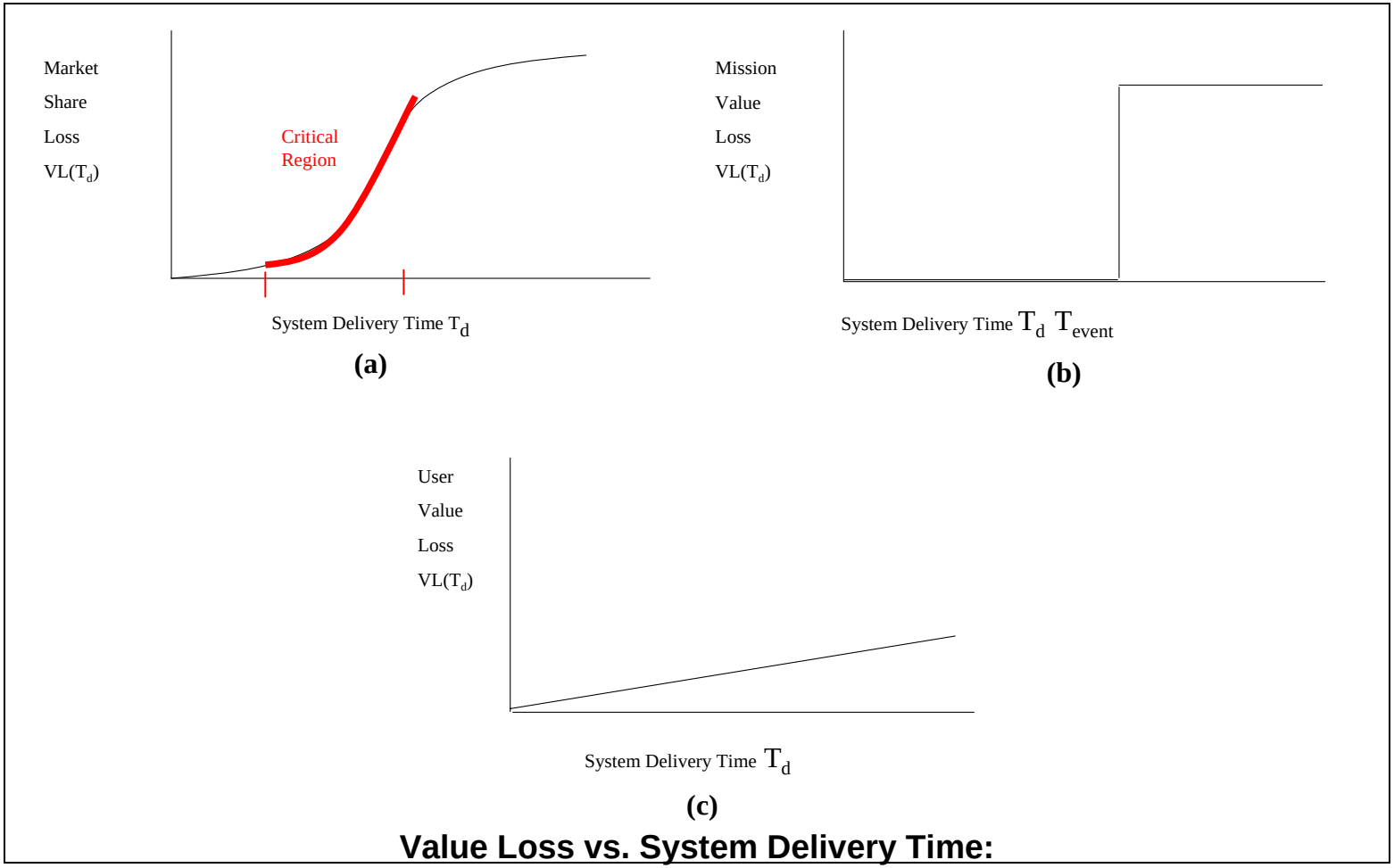
Source: LiGuo Huang, private communication.

Utility theory (for money)



Diminishing marginal returns

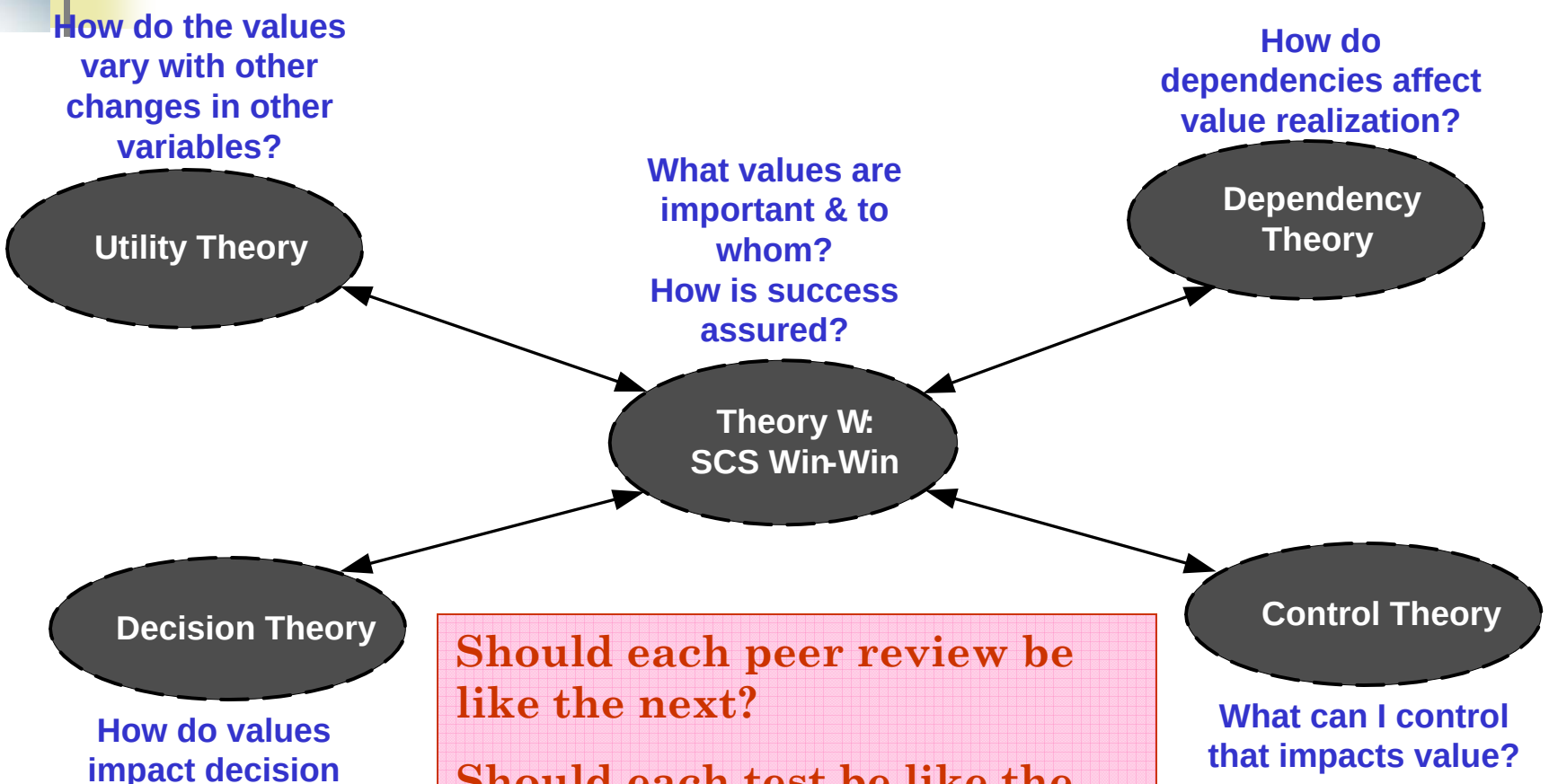
Other utility curves



Value Loss vs. System Delivery Time:
(a) Marketplace Competition (Internet Services, Wireless Infrastructure);
(b) Fixed-schedule Event Support; (c) Off-line Data Processing

Source: Huang & Boehm, op. cit.

4 + 1 Framework



Should each peer review be like the next?

Should each test be like the next?

Should each external & milestone review be like the next?