Value Creation with Software Process

Hakan Erdogmus
Kalemun Research Inc.

January 26, 2010

"Extreme Programming can create ten times more value than a heavy-weight process."

Kent Beck, XP-Universe 2001

- Which practices and characteristics could possibly make Extreme Programming economically superior?
- Why?

Outline

- The concept of value
- Tactical value
  - **Pair Programming**
- Strategic value
  - **Just-in-Time Decision Making**
  - **Iterative & Incremental Development**
- Putting it all together
  - **Test-Driven Development**
    - **Tactical (empirically based)**
    - **Strategic (speculative)**
- Summary

What's "value"?

- Economic Value

- (Net) Value = Customer Benefits - Developer Costs
- Minimize costs *or* maximize benefits
- Not always quantifiable
  - Uncertain – *ex-ante*, a statistical artifact
  - Time-dependent: $1M received now vs. $1M received in 5 years
  - Therefore:
    - Use models to make value tangible
    - Understand how value behaves under different assumptions
    - Get insights into modern development practices and processes

Sources of value in software process

Tactical – routine, systematic; uncertain benefits average out and accumulate over time;

- *Do it regularly to receive long-term benefits…*
- Cost effectiveness

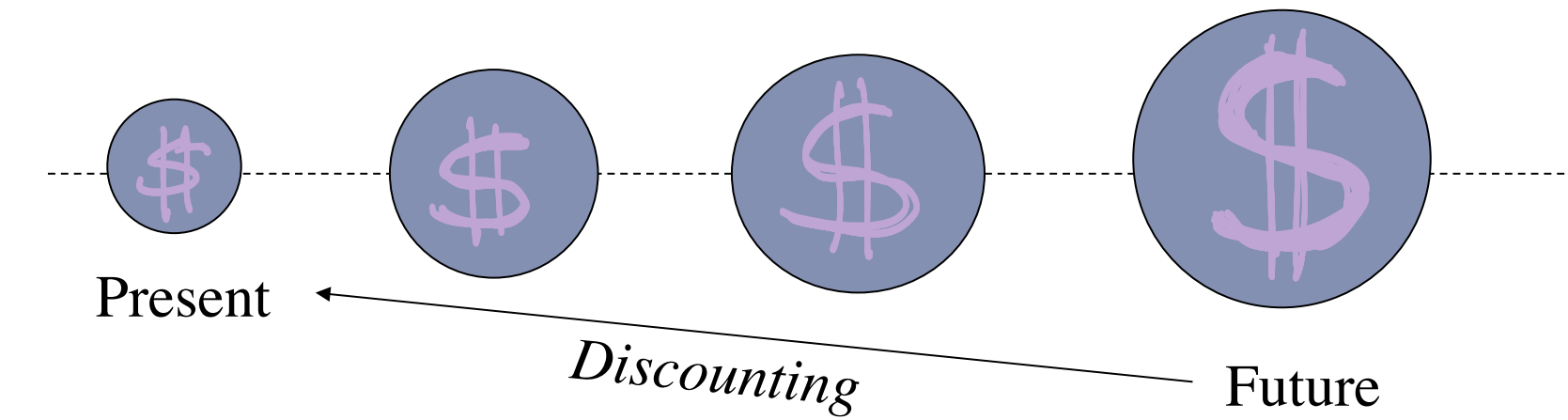
Strategic – situational or one-off, but planned; acted upon or materializes only under the right circumstances

- *Exercise the right strategies at the right time to maximize value…*
- Optionality

Structural - due to time value effects

- *mainly through incremental delivery (early benefits, deferred costs)*

Time value

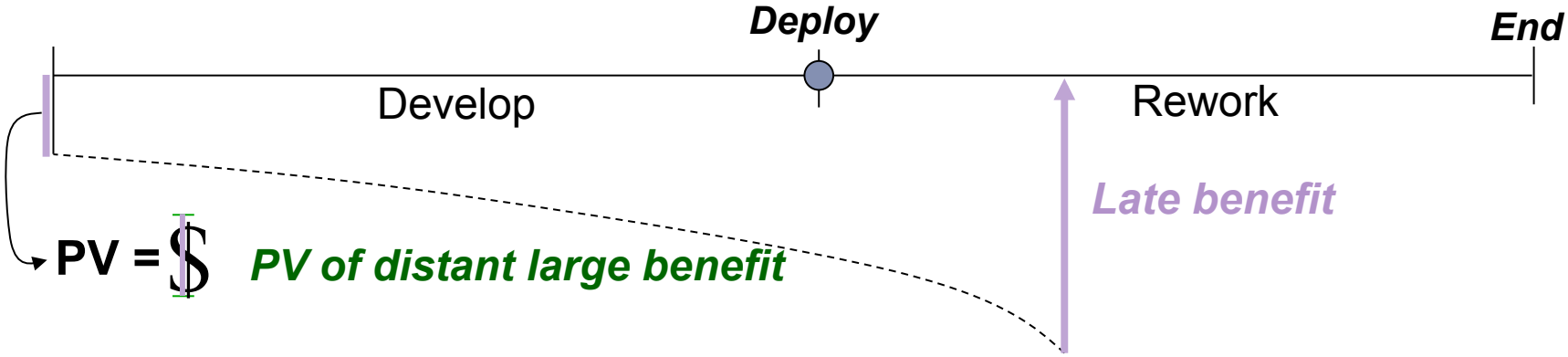


Earn early!

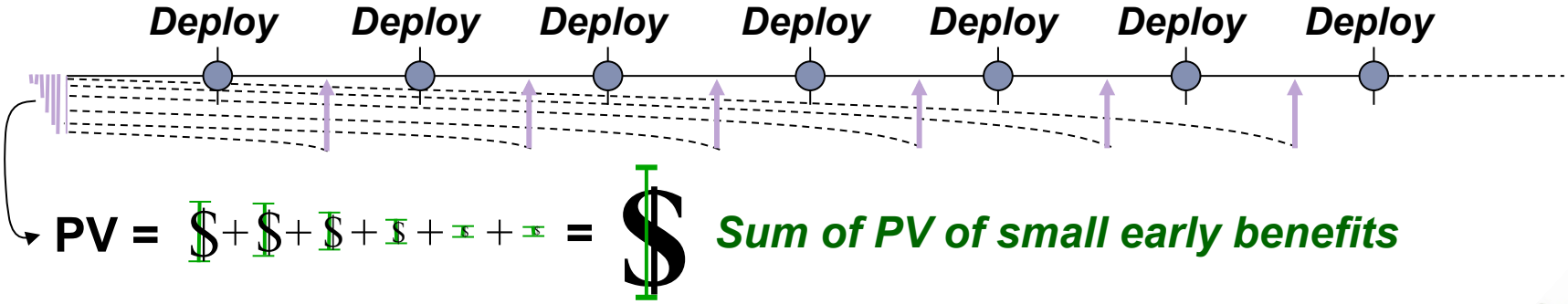
Spend late!

Time value favors incremental delivery

Infrequent -> Single-Point: Late benefits, all at once or in big chunks



Incremental -> Continuous: Early benefits, in small chunks



Tactical value: cost effectiveness

- *Doing the most and best in the shortest time possible*
- Usually independent of external uncertainty over the long term
- *Mainly* driven by cost savings/avoidance, productivity improvements

- Cost Effectiveness =

  Productivity [–] Cost of Poor Quality

- Productivity: output per unit time

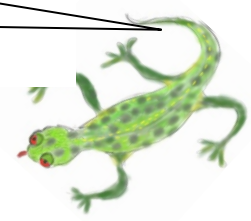
  (something to do with speed of production)
- Quality: lack of (a) defects that need repair, (b) issues that need addressing, or (c) actions that incur a deferred cost or prevent anticipated benefits from being realized

  (something to do with avoidance of rework and a prerequisite for usability)

Chair for Pair Programming

Do pair programming at your shop? These expensive Aeron's are for you:

A costly proposition?



Pair Programming – A tactical practice?

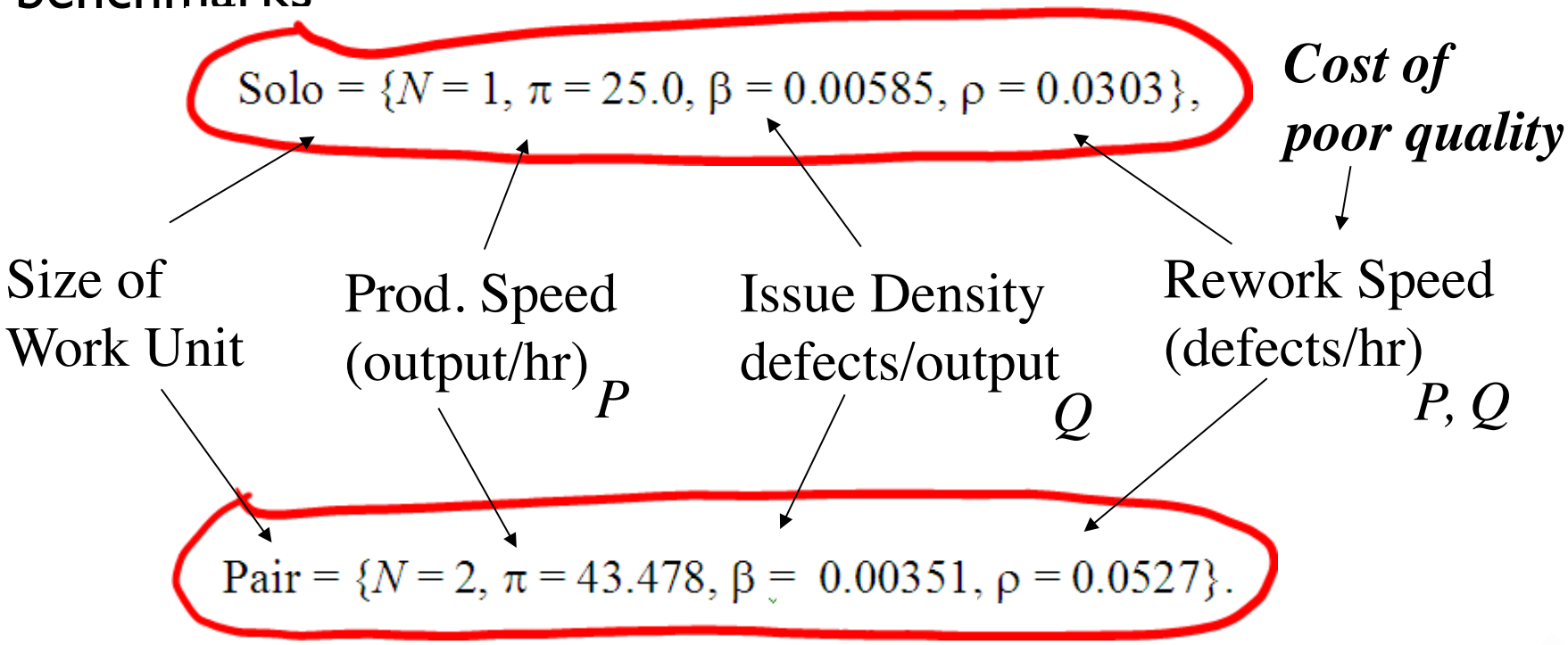
Analysis based on early empirical study by Laurie Williams comparing Solo and Pair programmers

- Pairs spent on average 15% more total effort than soloists to complete the same programming task
- Code written by pairs on average passed an additional 15% of the specified acceptance tests compared to code written by soloists
- Other studies report varying results; all claim PP is a costly proposition; most indicate some quality advantage

Early summary of evidence regarding PP: see "Voice of Evidence" in Nov/Dec 07 issue of IEEE Software

PP – Evaluation model

Results applied in simple model, supplemented by industry benchmarks



P: productivity *Q: quality*

PP value using an economic indicator

Is PP really that costly based on these observations?

- Total Benefit = (Benefit per Unit of Output) x (Output)

Unit Value (UV)

- Net Value (NV) = Total Benefit – Total Cost ?

Measure of Profitability

- Breakeven Unit Value (BUV) = min { UV | 0 ≤ NV }
  - *The smaller, the better!*
  - *BUV is inversely proportional to real productivity*

- Compare BUVs for Pair and Solo under continuous delivery

PP cost effectiveness results

BUV Ratio

$$\textbf{BUVR} = \frac{\textbf{BUV(Solo)}}{\textbf{BUV(Pair)}}$$

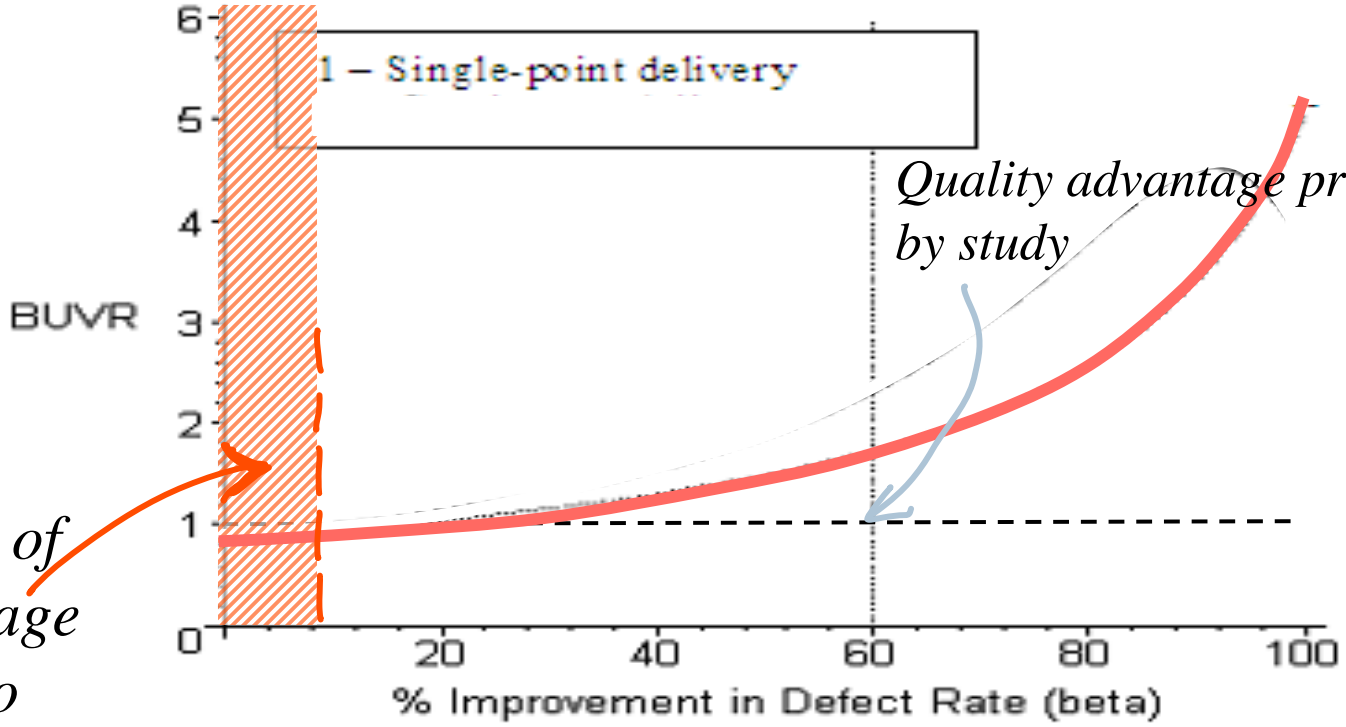
$> 1 \Rightarrow$ Pair ✓

$< 1 \Rightarrow$ Solo ✓

	BUVR Behavior	
Condition \ **Value Realization Model**	**Single-Point Delivery**	**Continuous Delivery**
Discount rate (r) increases } TIME VALUE	BUVR increases	BUVR = 1.73
$r \rightarrow 0$ } TIME VALUE	BUVR $\rightarrow$ 2.24	
Output (ω) increases	BUVR increases	
Overall better model	Pair ✓	Pair ✓

PP – Sensitivity analysis

BUVR most sensitive to quality



PP is a quality technique?

Optionality

- Optionality = Learning [+] Flexibility
- Learning: awareness of how uncertainty is resolved
- Flexibility: taking advantage of learning outcomes
- Ability to respond to change (take an appropriate action) in an manner that maximizes value given the information available at the time when the opportunity to take the action presents itself
- Matters only under uncertainty
- *Mainly* driven by opportunistic benefits maximization

Uncertainty, learning, flexibility

- No learning $\Rightarrow$ Flexibility is useless

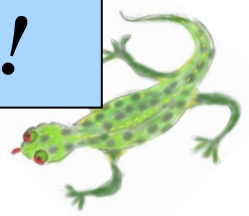
  No difference between deciding now or later, thus might as well make the decision now

- No flexibility $\Rightarrow$ Learning is useless

  No choice any way, so why bother?

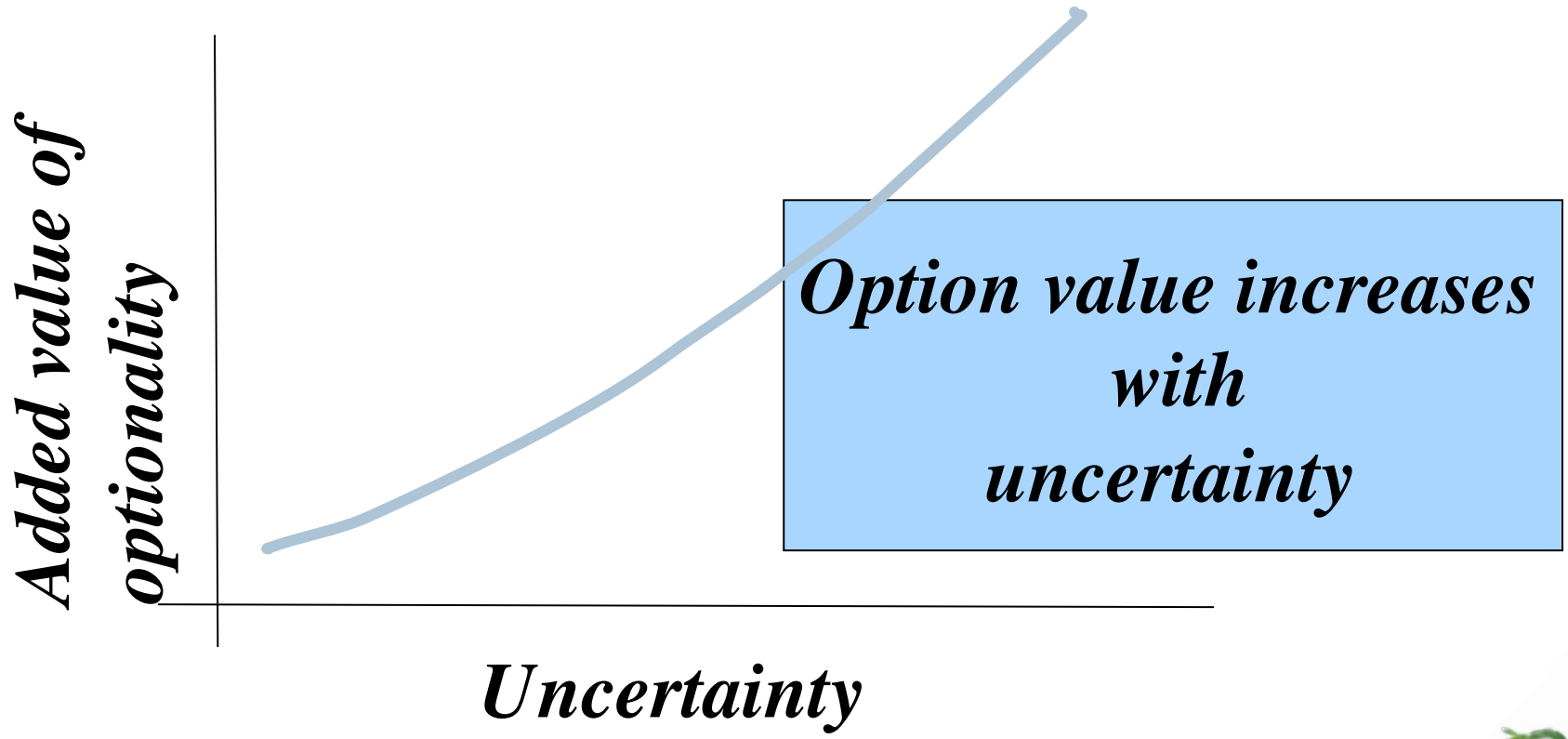
- No uncertainty $\Rightarrow$ No need for flexibility, nor learning

Optionality is valuable only under uncertainty!



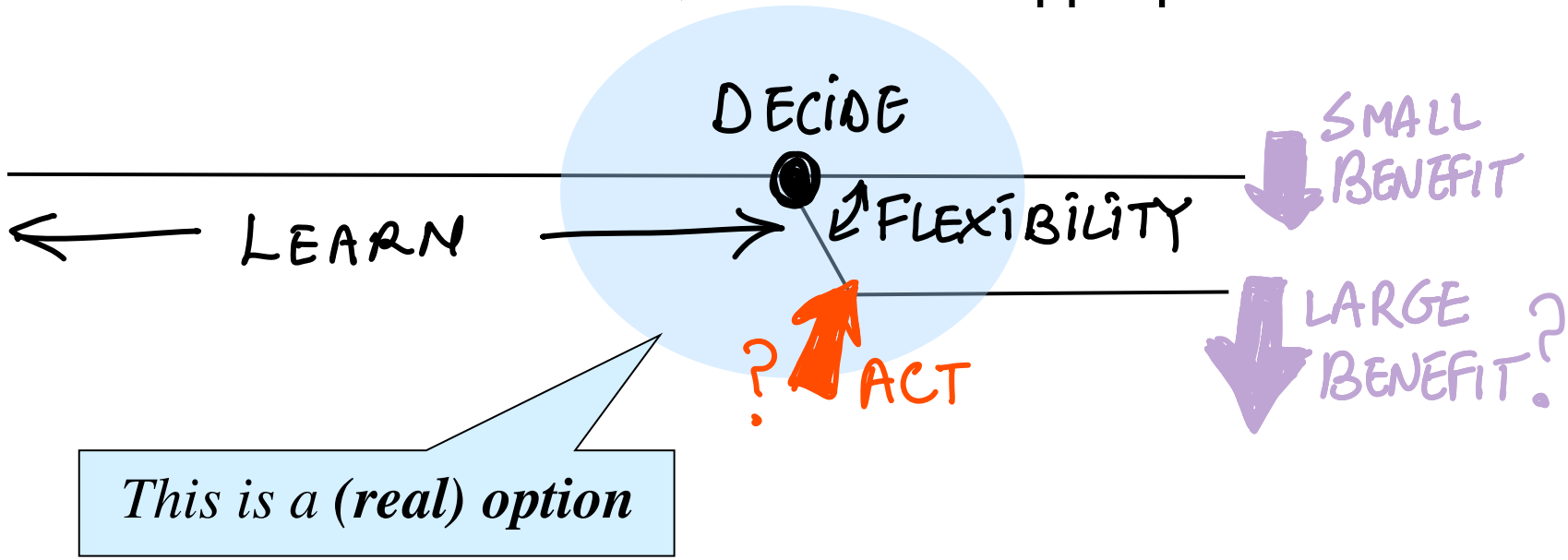
Uncertainty & optionality

Value added by optionality most sensitive to uncertainty!

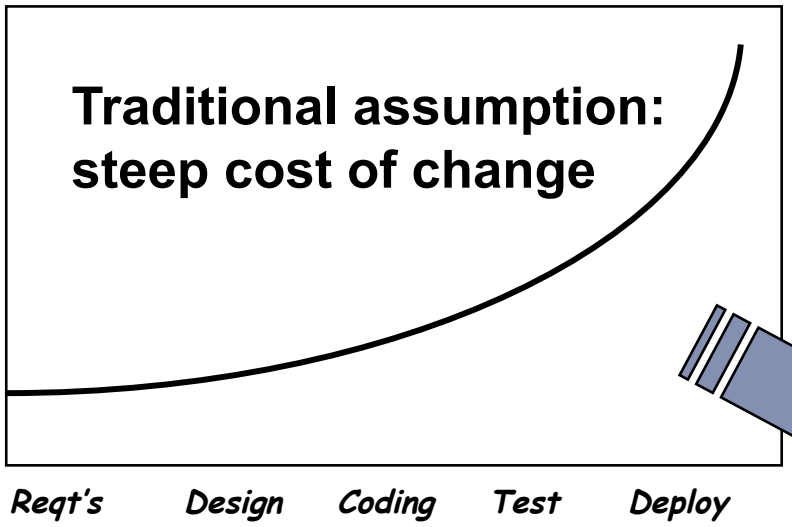


Reactive optionality

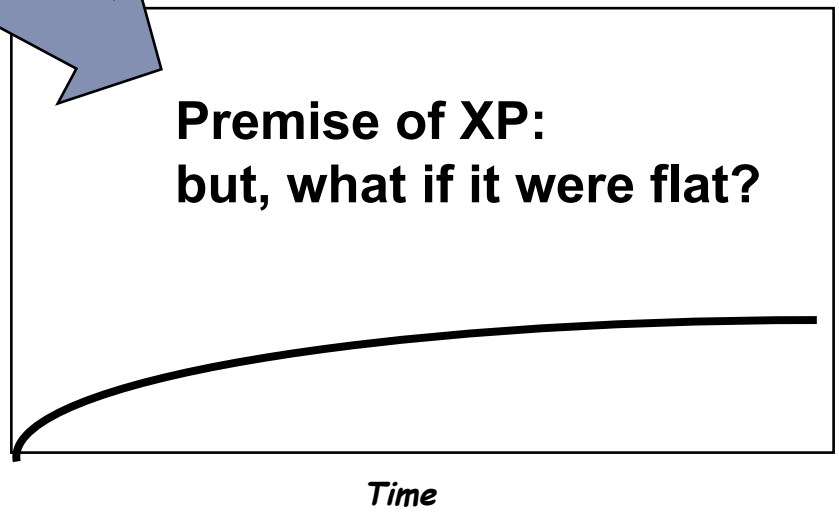
- Do nothing now
- Wait for new information, react when appropriate



Reactive optionality: Just-in-time decision making



***Development decisions under* different cost of change curves**



Case for early decisions/actions (E.g. BUFD)

Early decisions & preventive actions are better!

Deferred decisions/actions are better!

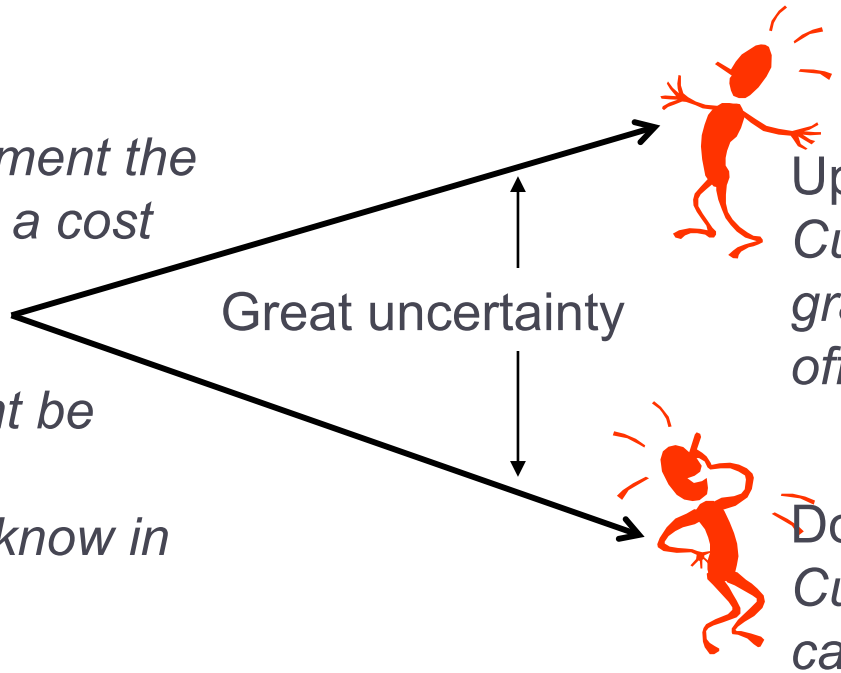
Case for JIT decisions/actions (E.g. YAGNI)

An example: YAGNI (You Ain't Gonna Need it)

You are thinking of implementing a new feature…

You could implement the feature today, at a cost of $1000

You think it might be worth $1500 in benefits... We'll know in a year!



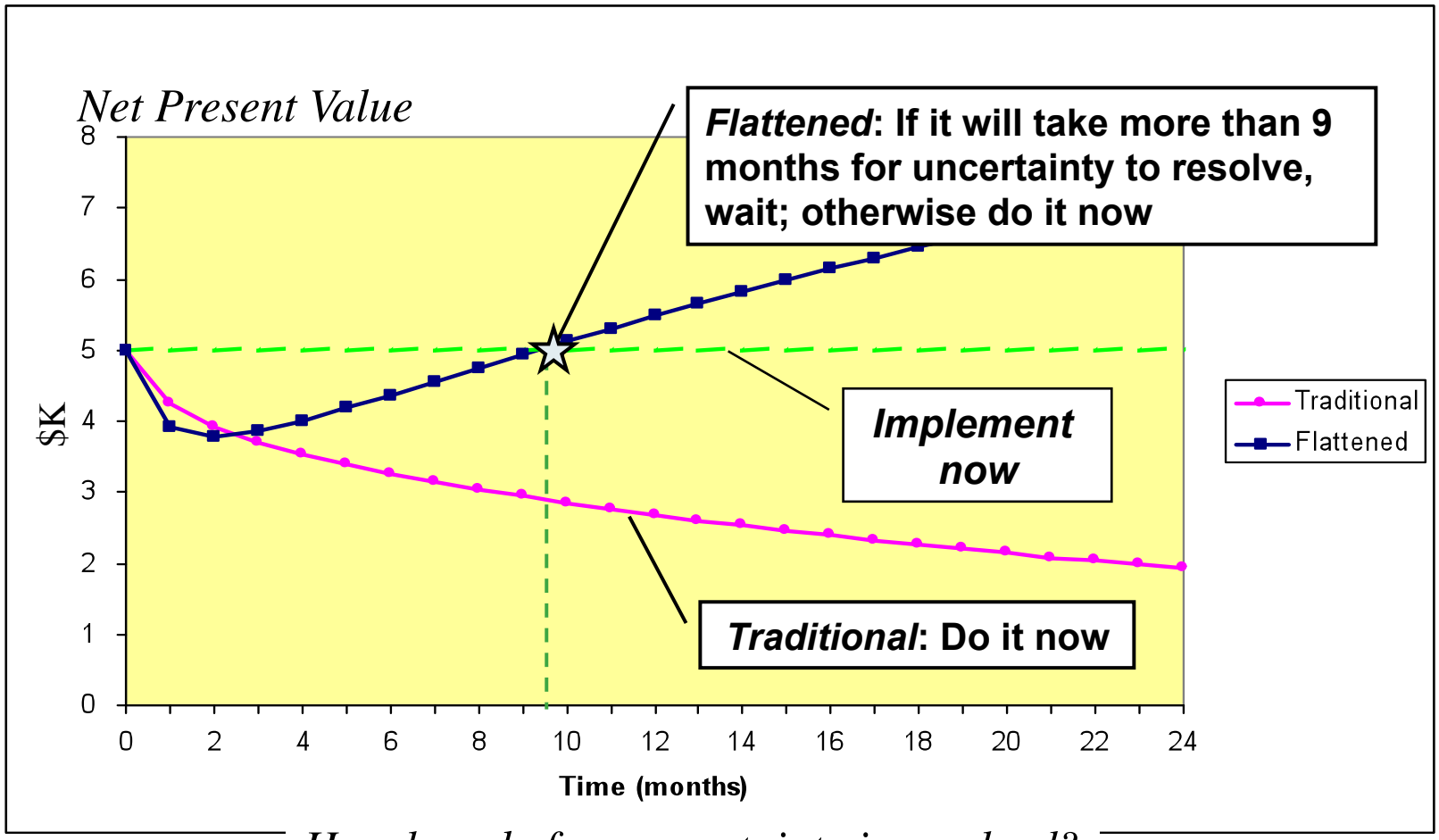
Upside scenario: *Customer is grateful; it all pays off*

Downside scenario: *Customer couldn't care less—you wasted time and money*

Courtesy of John Favaro, 2000

Value of JIT decision making as a real option

Effect of change cost and time horizon

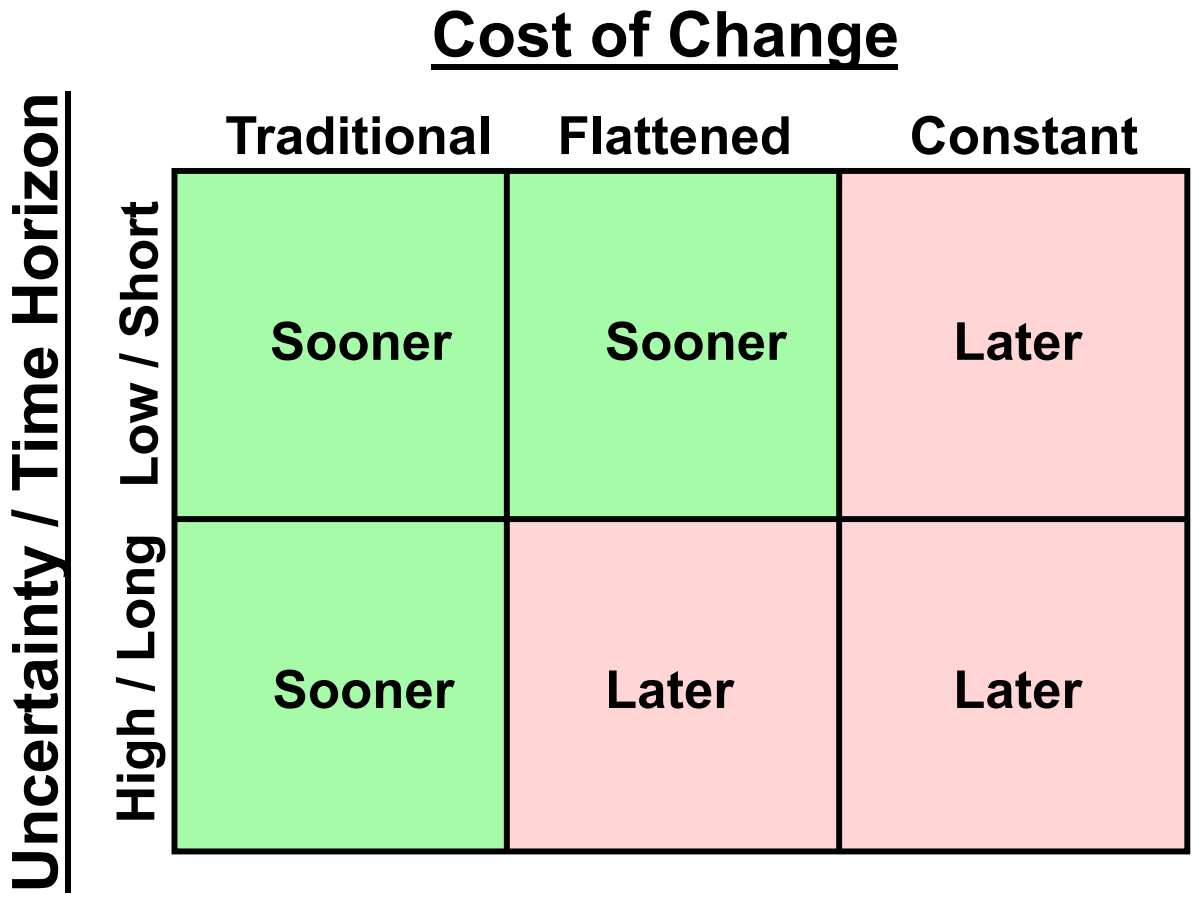


How long before uncertainty is resolved?

Rules of thumb for JIT decision making

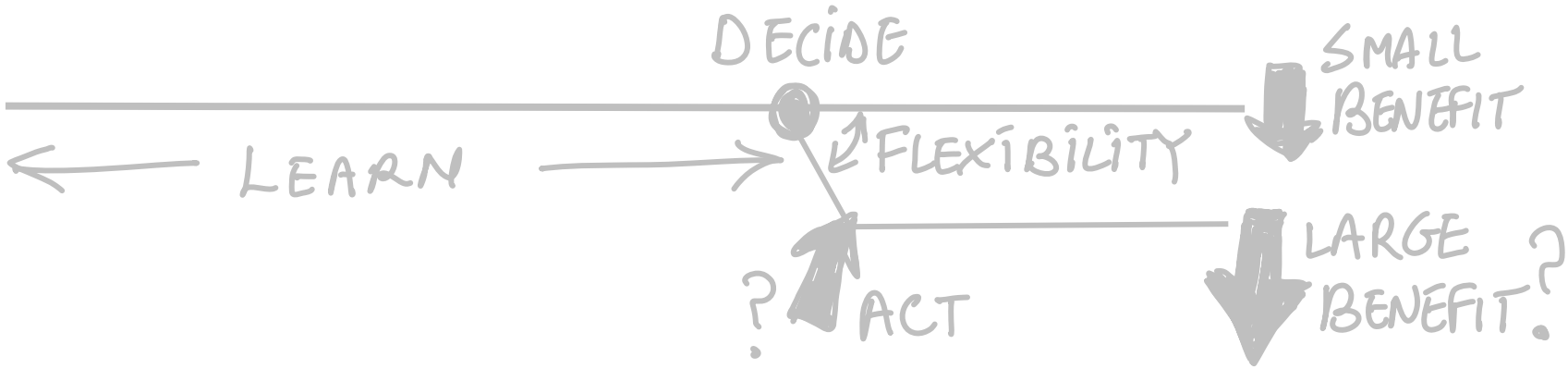
… based on real-options-based value analysis

… relative to change cost and benefits uncertainty

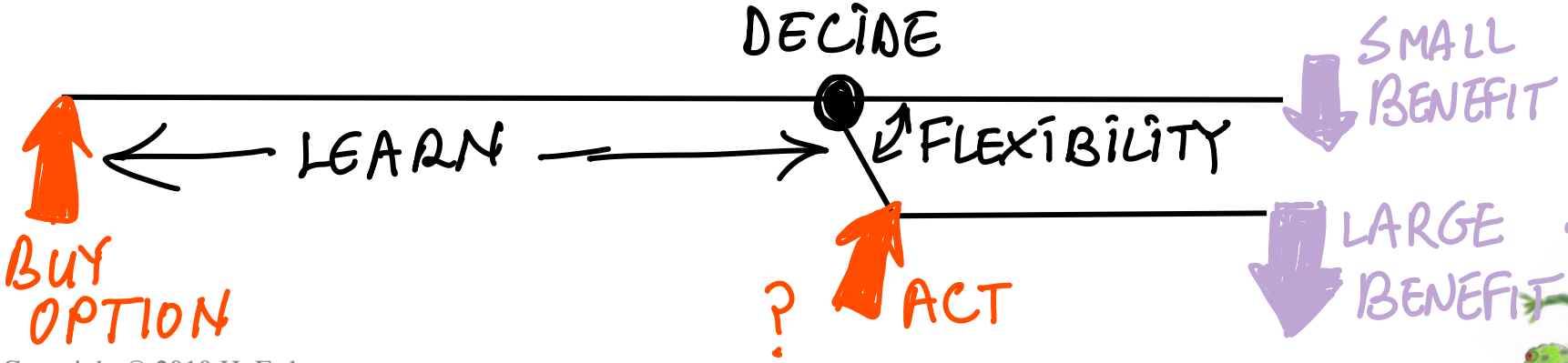


Proactive optionality

- Reactive



- Proactive: do something now to be able to react later

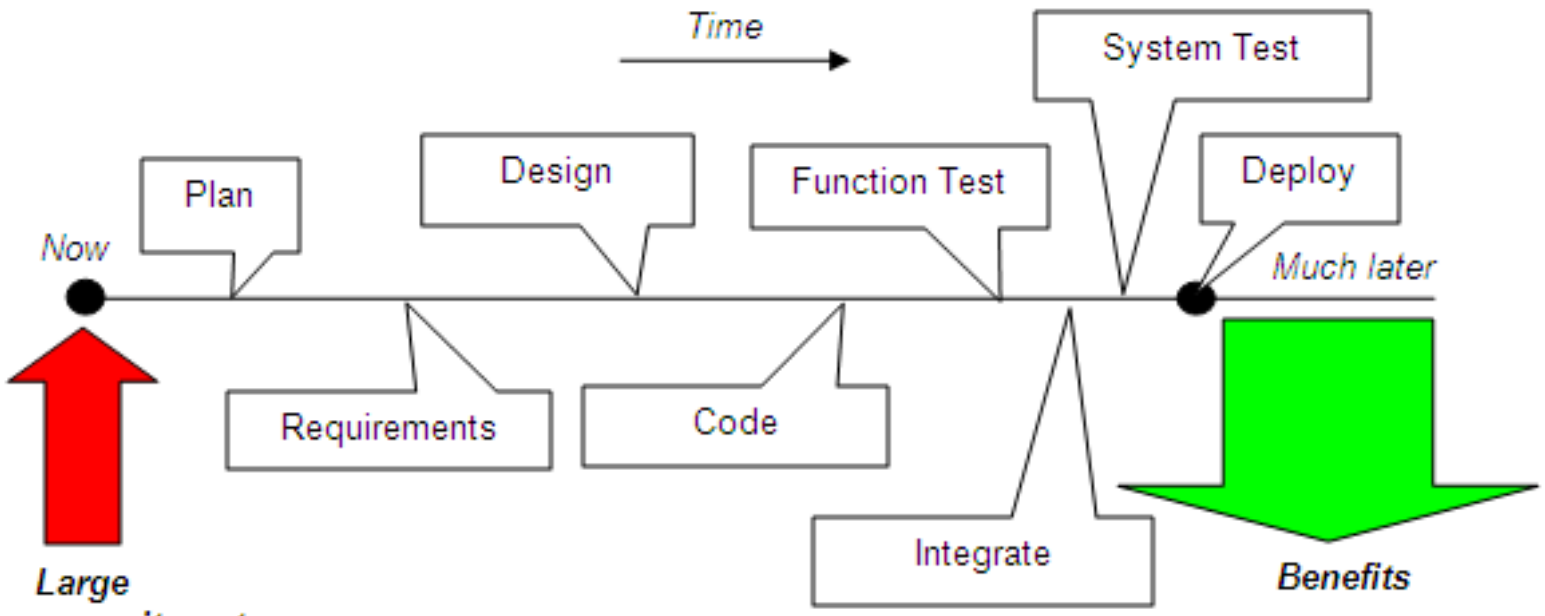


Iterative & incremental development

- Incremental: *Deliver usable functionality in chunks*
- Iterative: *Repeat essentially the same process*

- *Incremental development has added value due to time value effects (already discussed)*
- *Iterative development has strategic value due to optionality*

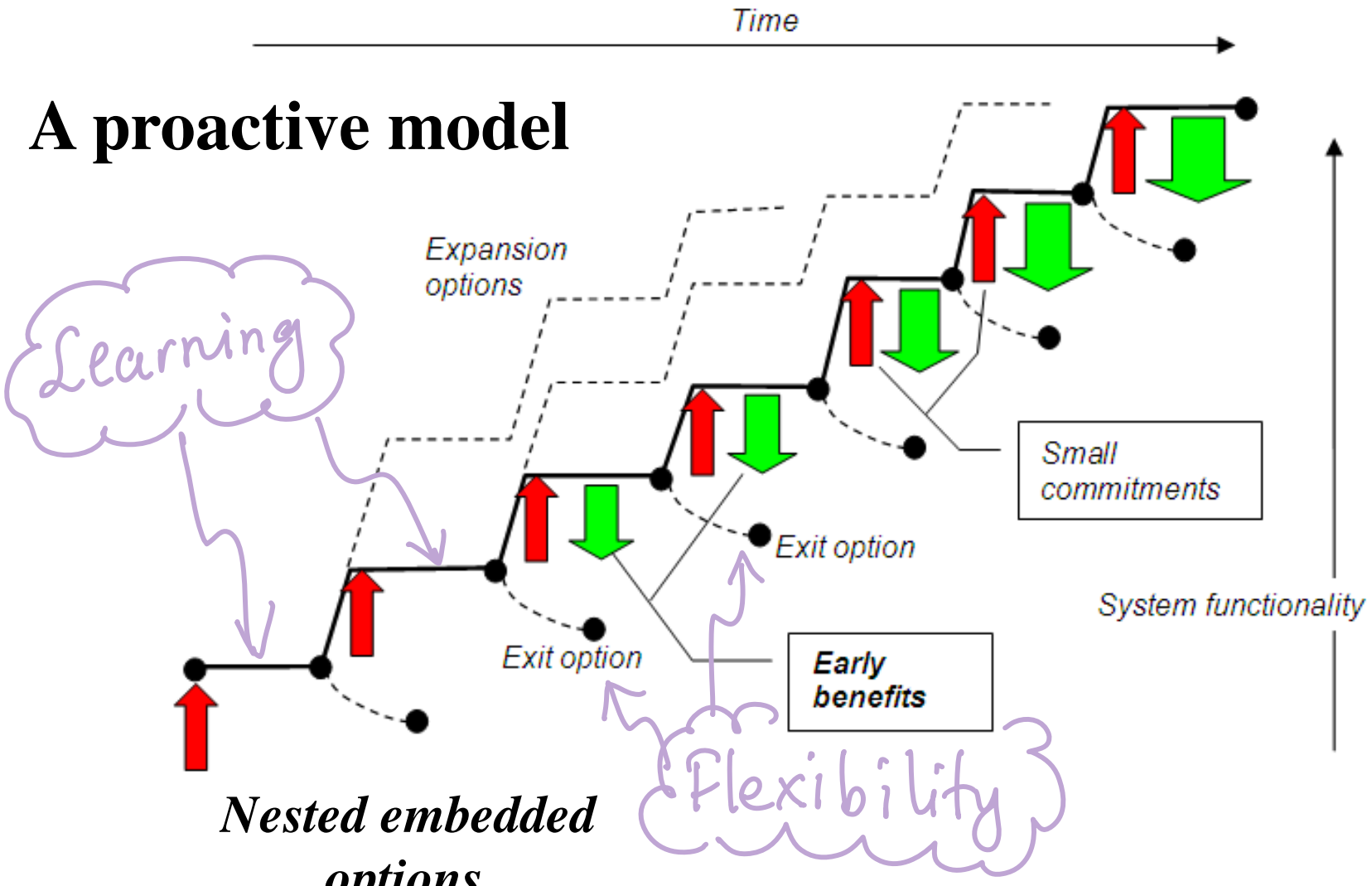
Economics of sequential development



(upfront)

(late)

Economics of IID



Learning power

- The capacity of a project stage to resolve the project's uncertainty

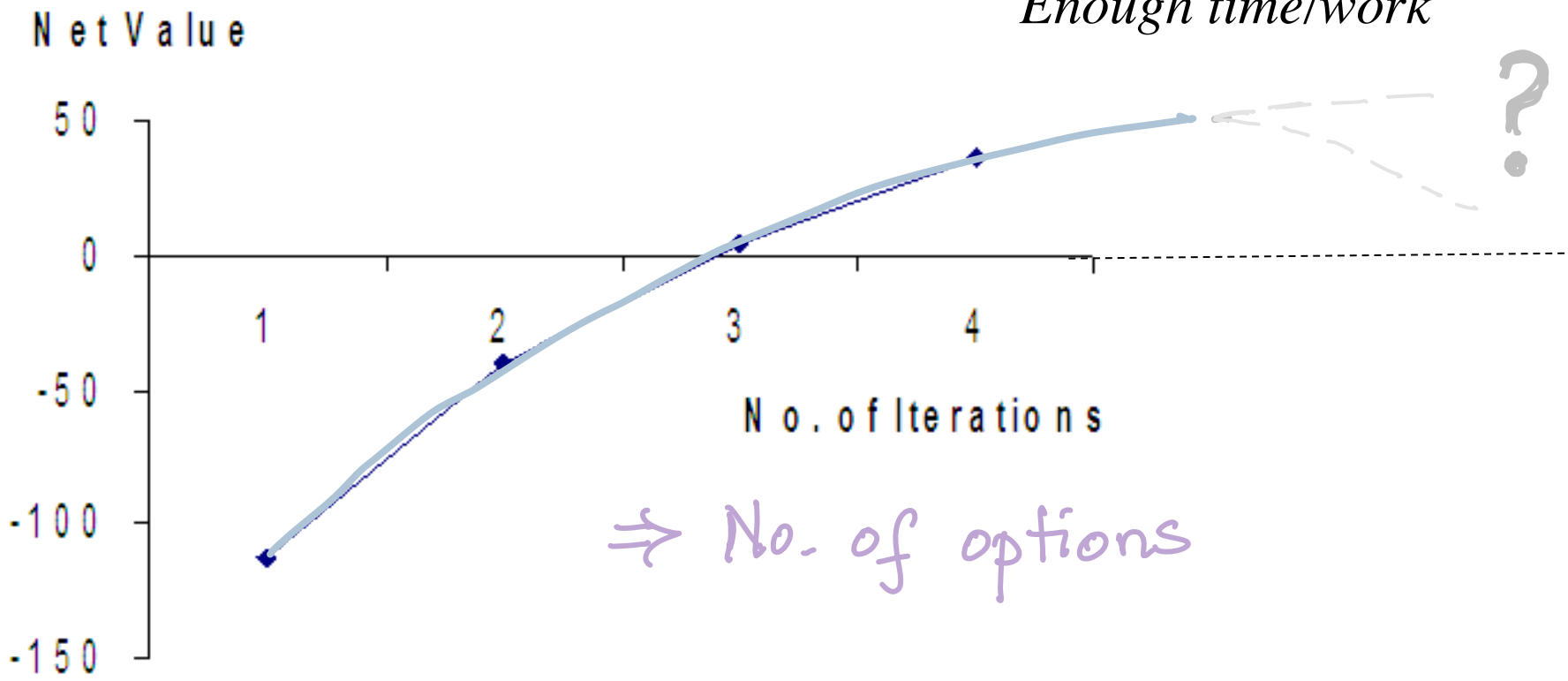
- Each successive iteration decreases the variance of project's benefits based on the iteration's outcome
  - Good outcome => Prob. of good outcome increases
  - Poor outcome => Prob. of good outcome decreases
  - Stage-gate processes work like this: e.g., pharmaceuticals development, venture capital, natural resources development

- Learning power ⇑ ⇒ Value of staged project ⇑

Tackle *show stoppers* and *proof of concepts* early!

Process granularity

Fixed overhead
Enough time/work



Keep iterations as short as practicable!

Relative tactical value of TDD

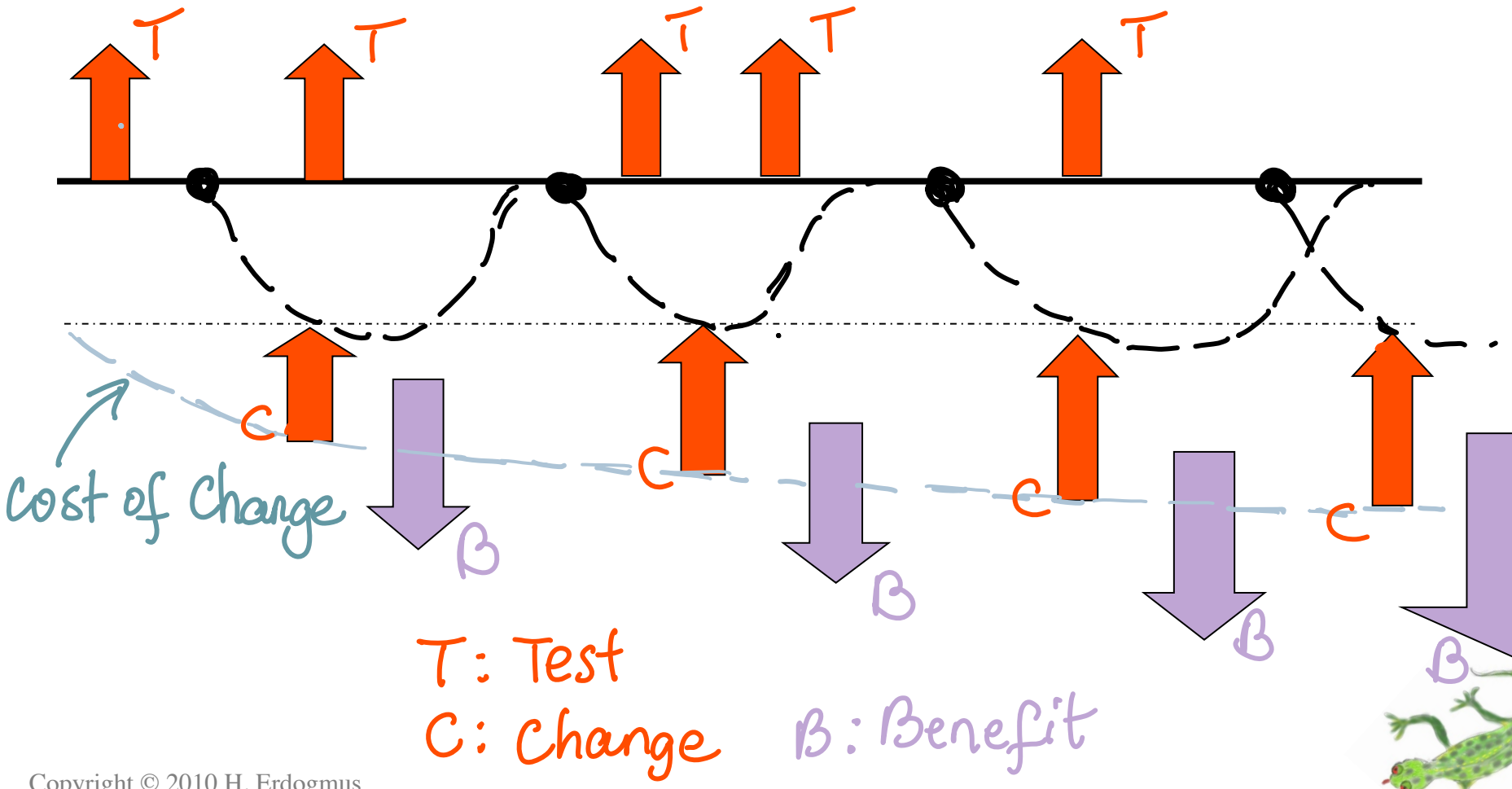
Early empirical evidence:

- Compared with a technique in which testing is not mandatory: mainly a *quality* technique
  - Should not be surprising
- Compared with a technique in which testing is mandatory: mainly a *productivity* technique

Strategic value of TDD: option value of test assets

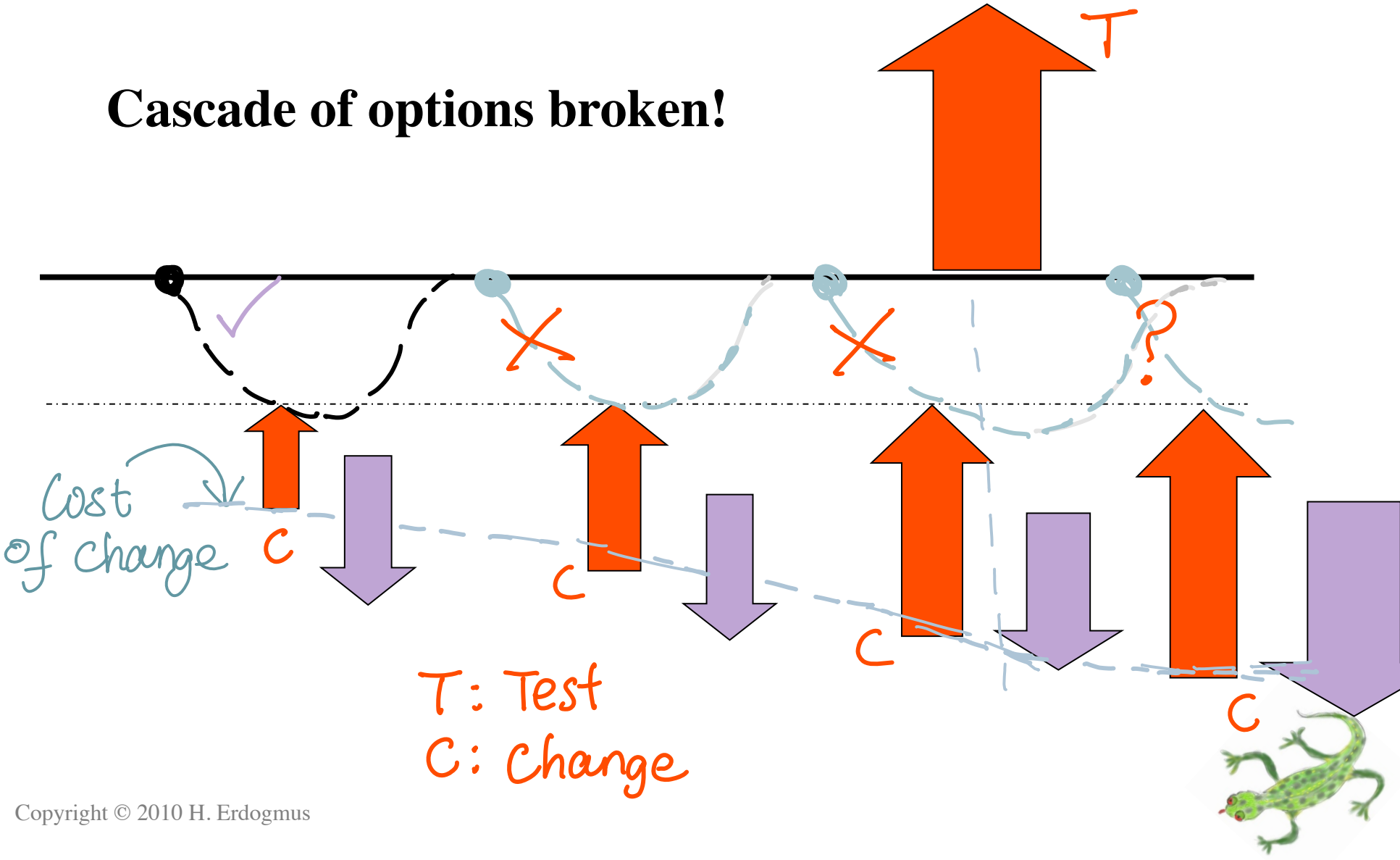
Proactive Optionality Model

TDD creates cascade of change options!

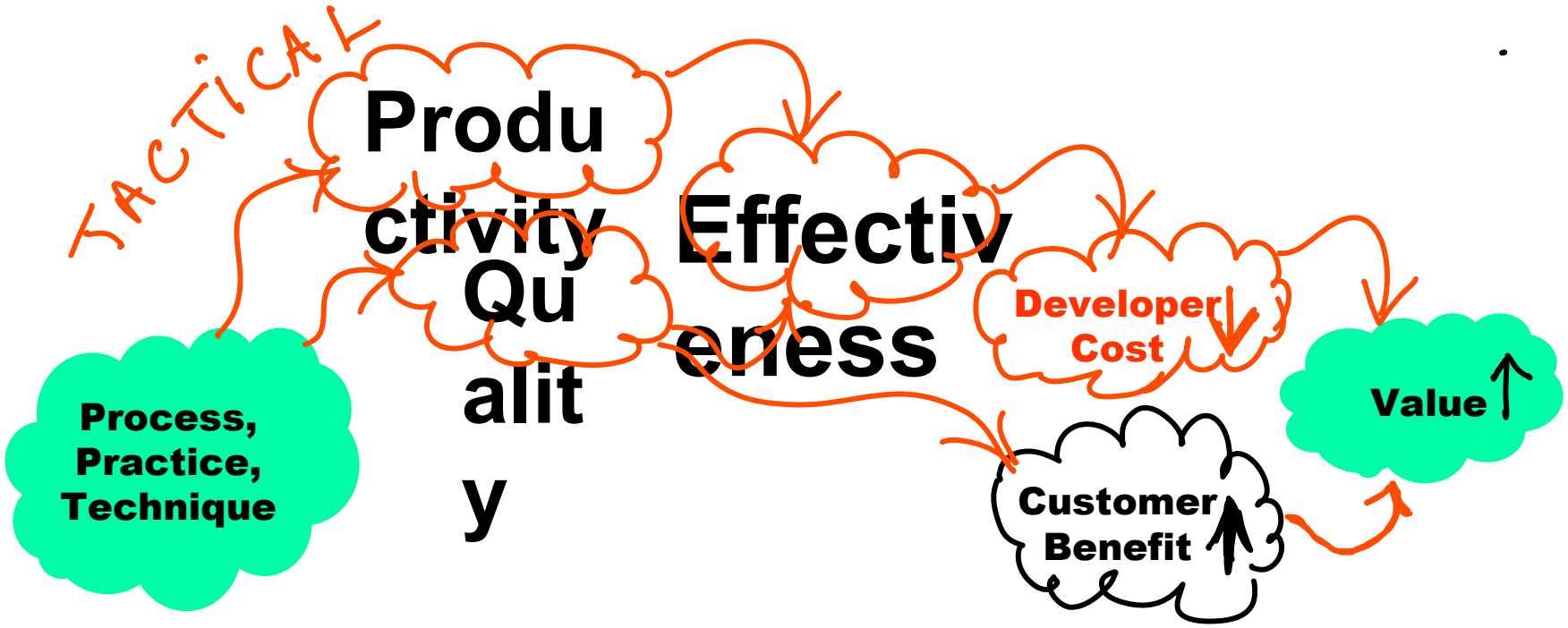


Option value may significantly diminish with after-the-fact testing!

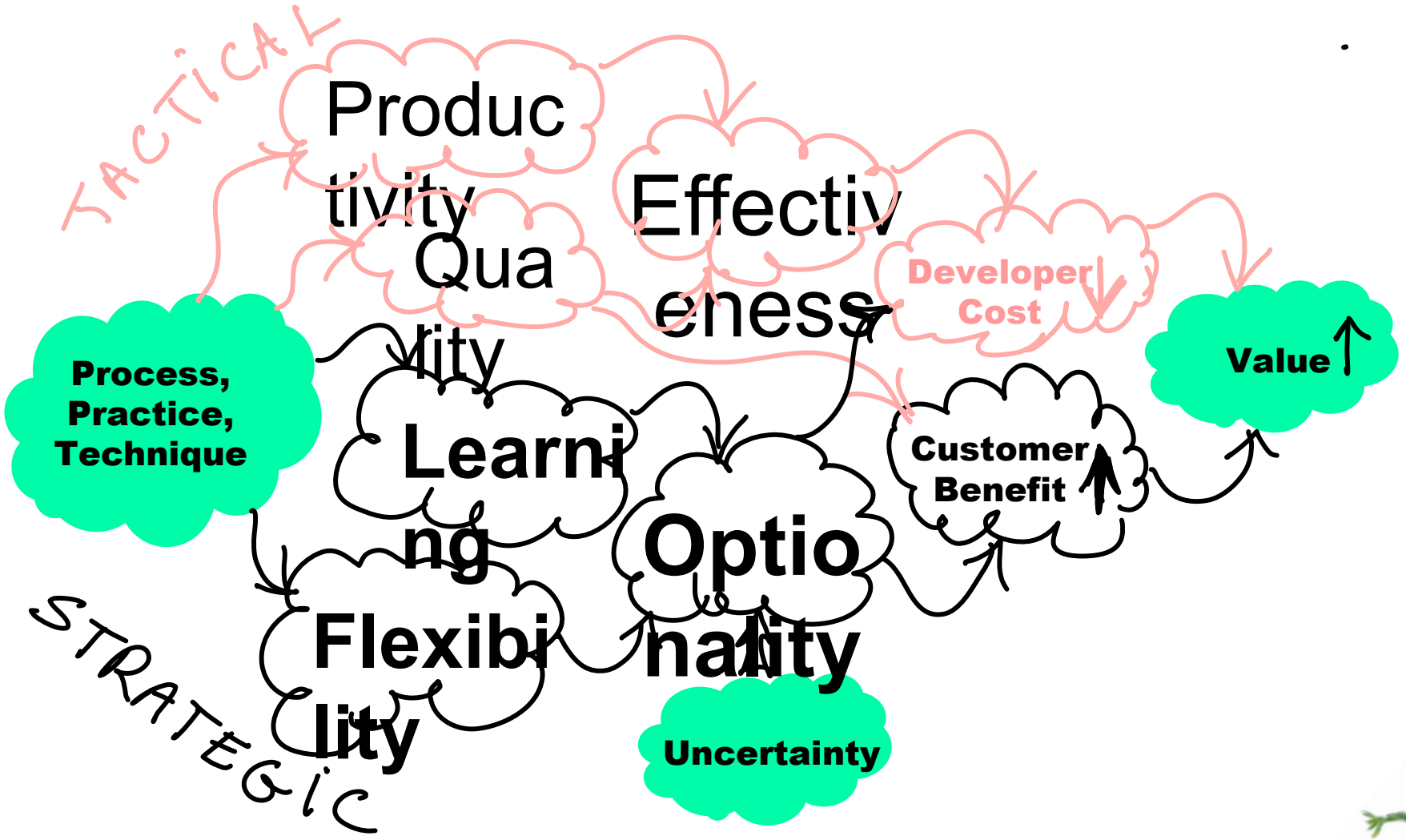
Cascade of options broken!



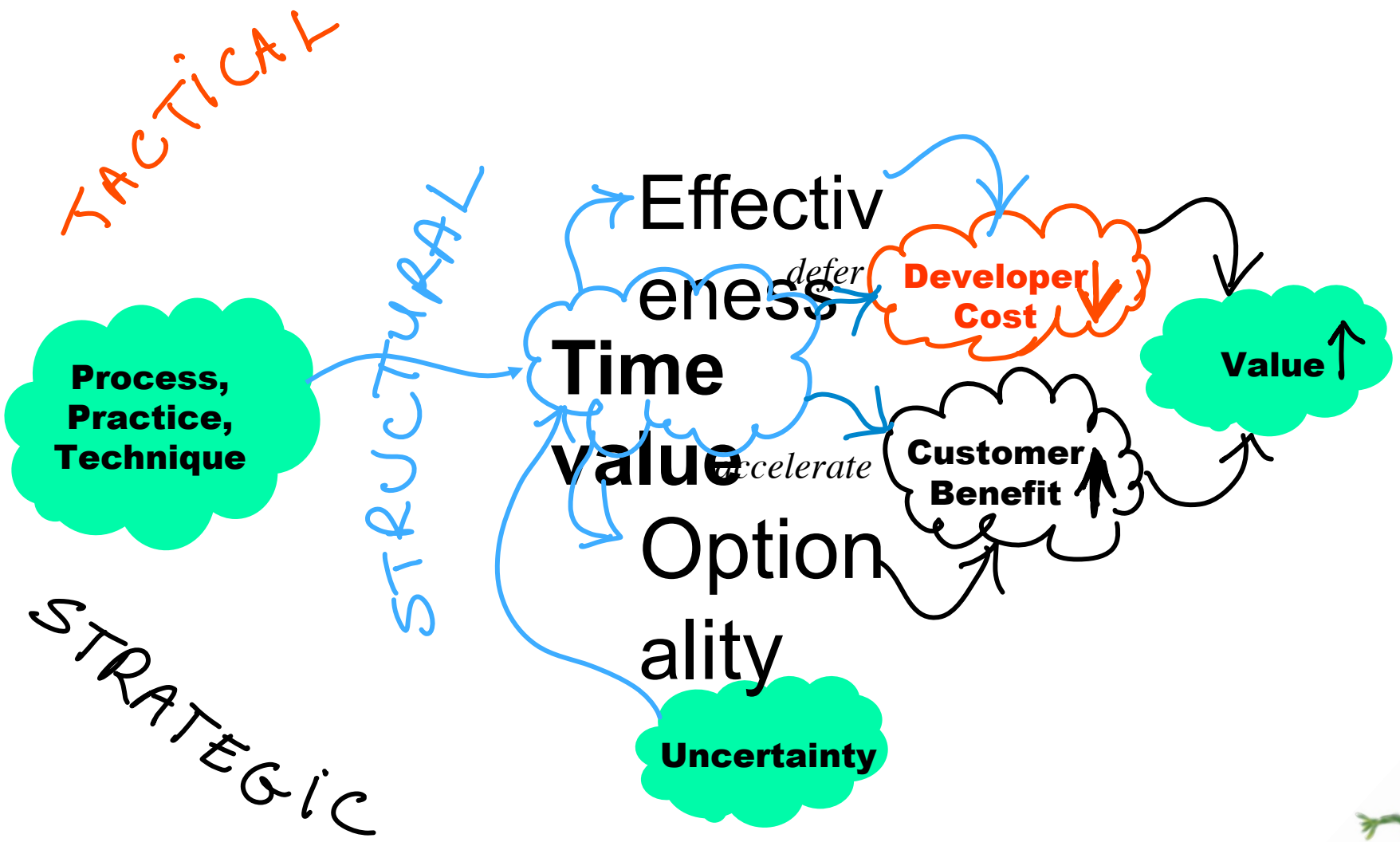
A value creation framework for software process



A value creation framework for software process



A value creation framework for software process



Related Reading

- B. Boehm & D. Turner. *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley, 2003
- H. Erdogmus. The Impact of Learning and Flexibility on the Economics of Iterative Development. *IEEE Software*, 2005
- H. Erdogmus & J. Favaro. Keep Your Options Option: Extreme Programming and Economics of Flexibility. In *Extreme Programming Perspectives*, Addison-Wesley, 2002
- H. Erdogmus and L. Williams. The Economics of Software Development by Pair Programmers. *Engineering Economist*, 48(4), 2003
- H. Erdogmus, M. Morisio, M. Torchiano. On the Effectiveness of the Test-First Approach to Programming. *IEEE Trans Software Eng*, 31(3), 2005
- Shull et al. Are two heads better than one. IEEE Software, Nov/Dec 2007
- H. Erdogmus. A cost-effectiveness indicator for software development. *Int'l Symp Empirical Software Eng & Measurement* (ESEM), 2007