

Bug Bash by Hans Bjordahl

<http://www.bugbash.net/>

# Process, Risk, and Scheduling



www.dilbert.com

©2008 Scott Adams, Inc./Dist. by UFS, Inc.

# One-Slide Summary

- A **software development process** organizes activity into distinct phases (e.g., design, coding, testing, etc.). Processes can increase **efficiency**, but are often implemented poorly.
- **Effort estimation** is based on historical information (models, experience). It is complicated by **uncertainty**, which stems from **risk**, which can be **managed** (identified, minimized). A project **plan** (milestones, deliverables) includes all of these considerations. Measuring progress is difficult.

# Process

- A **software development process** (also known as a **software development life cycle** or **software development model**) divides software development into distinct phases to improve design, product and project management.
- Process is “the set of activities and associated results that produce a software product”.
- Examples include the waterfall model, spiral development, agile development and extreme programming.

# Richard Feynman's Problem Solving Algorithm

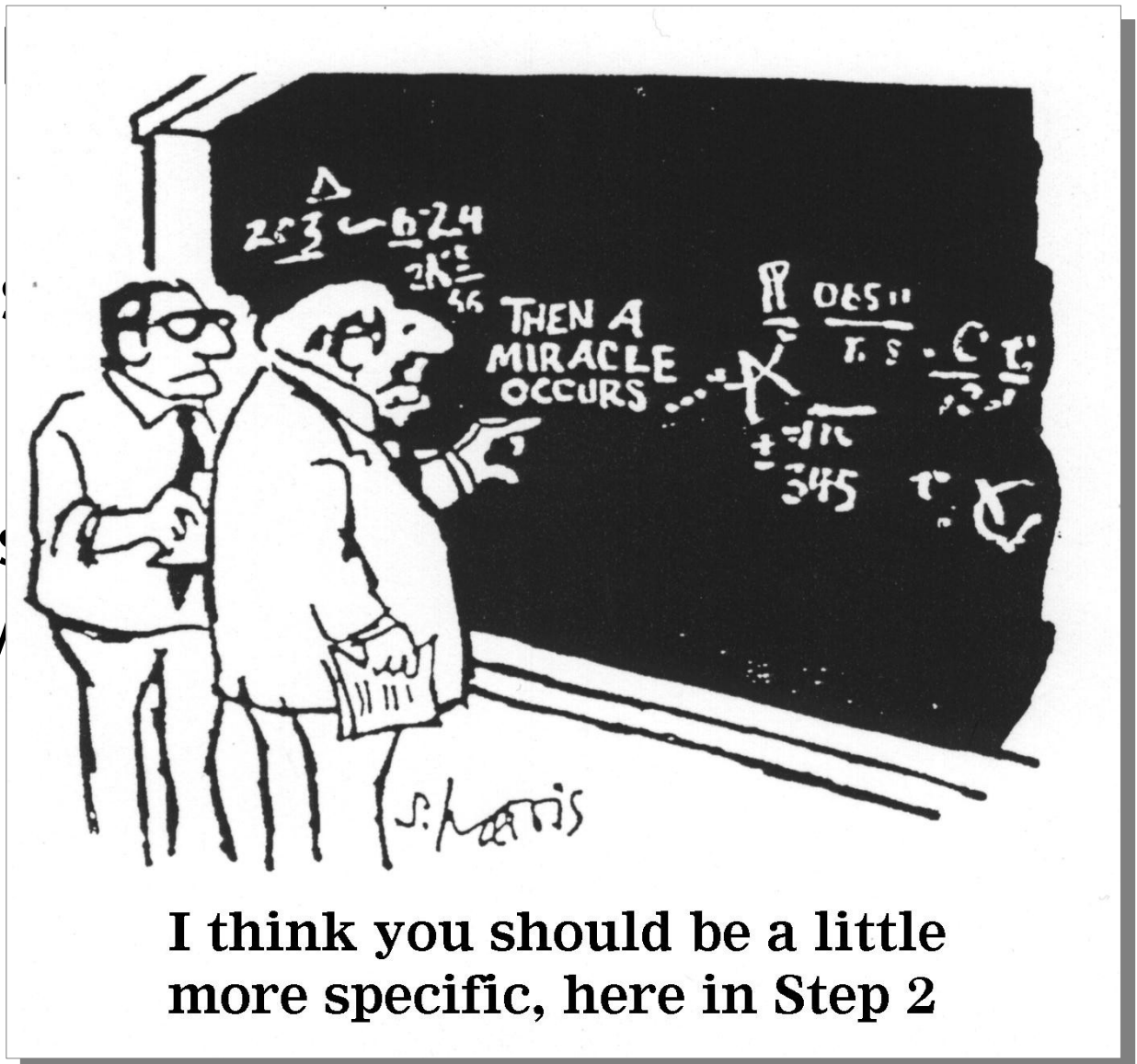
1. Write down the problem.
2. Think real hard.
3. Write down the solution.

- As facetiously suggested by Murray Gell-Mann, a colleague of Feynman, in the New York Times

# Richard Feynman's Problem Solving Algorithm

1. Write down the
2. Think real hard.
3. Write down the

- As facetiously  
colleague of Fey



# A Straw Software Process

- Discuss the software that needs to be written
- Write some code
- Test the code to identify the defects
- Debug to find causes of defects
- Fix the defects
- If not done, return to Step 1

chrwei • 59m

I once had complaints that a process was taking too long. no way to make it faster without gutting the whole system, so i added a progress bar, which actually made it take 5% longer, but the complaints stopped.



Reply

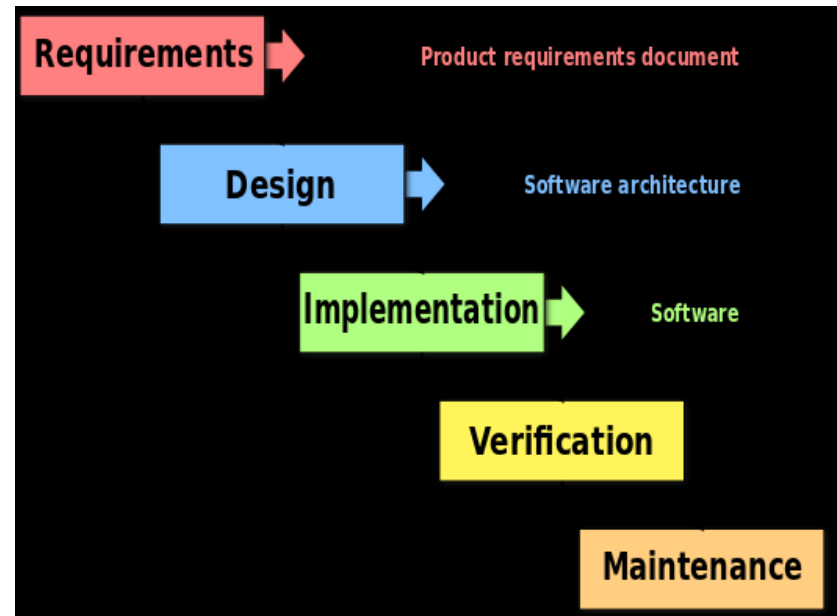
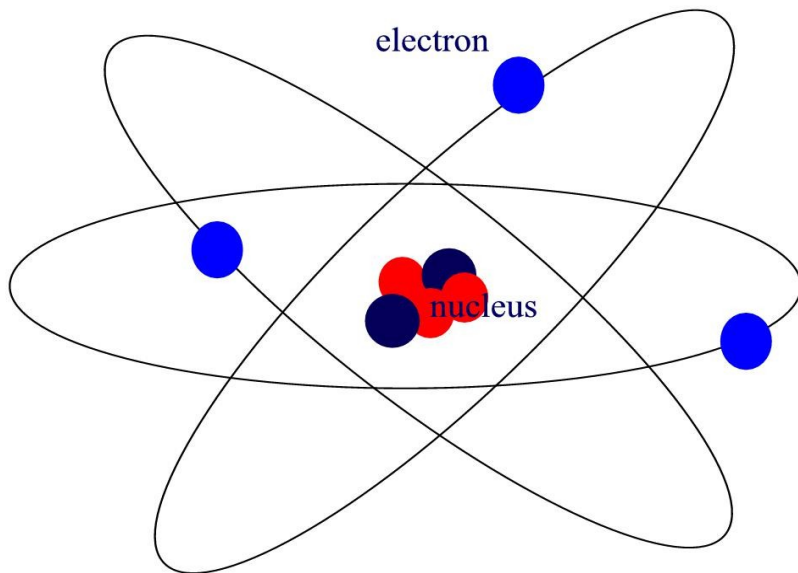
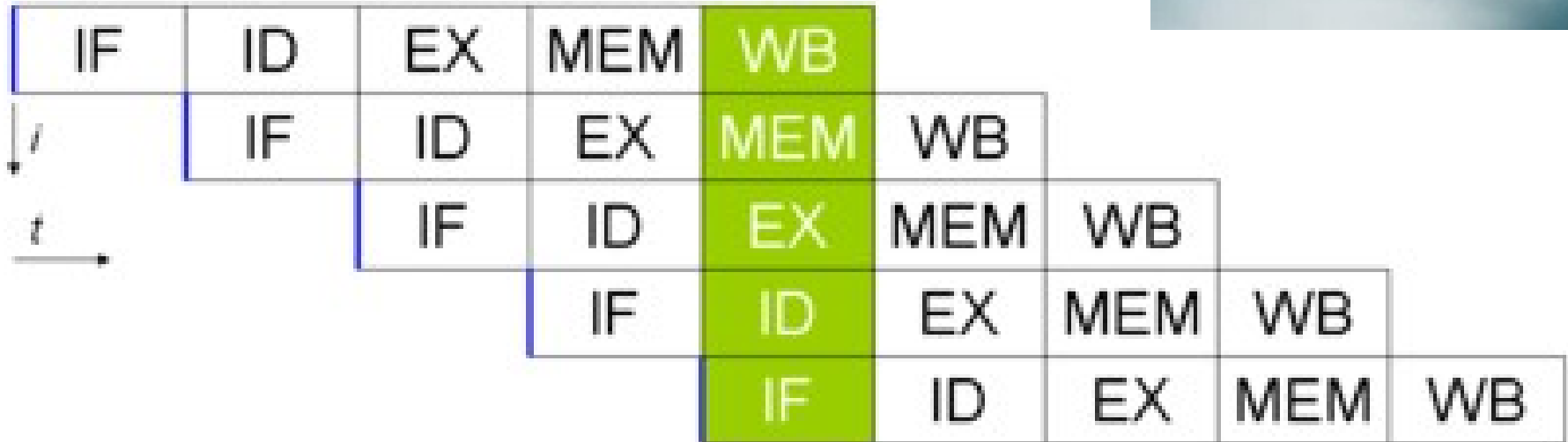
Vote

# Waterfall Model



- In the **waterfall** software development model, the following phases are carried out *in order*:
- System and software requirements: captured in a document
- Analysis: resulting in models, schema, and business rules
- Design: resulting in the software architecture
- Coding: the development, proving, and integration of software
- Testing: the systematic discovery and debugging of defects
- Operations: the installation, migration, support, and maintenance of complete systems

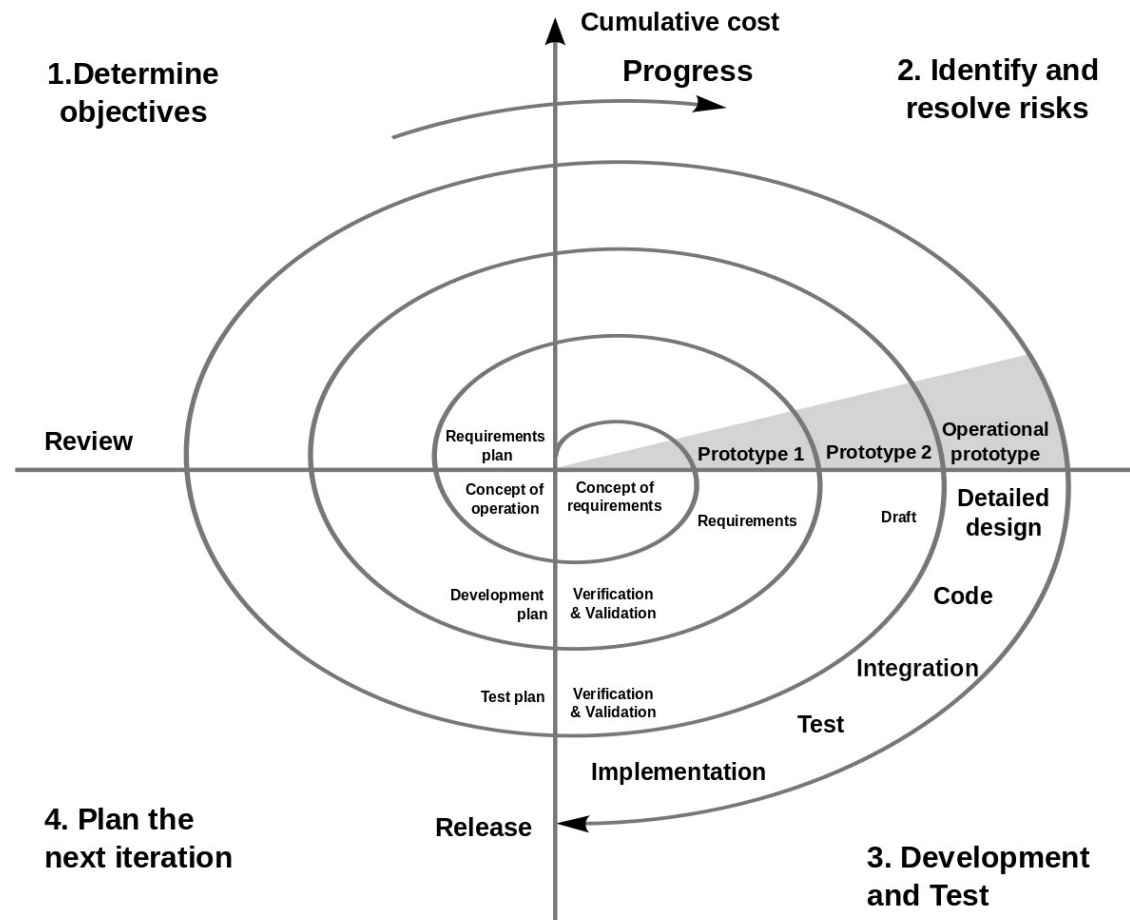
# Tell Me Lies



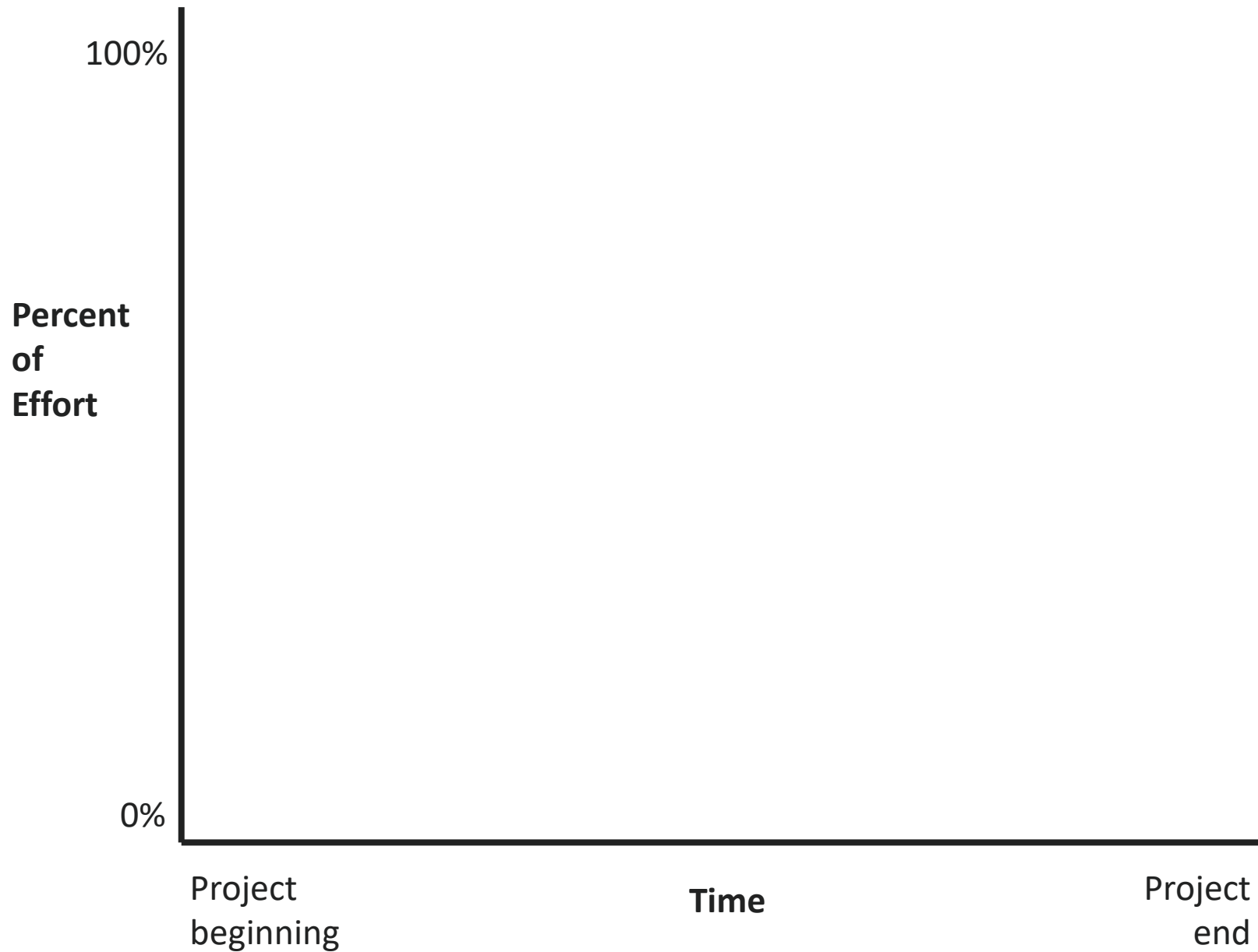


# Spiral Development Model

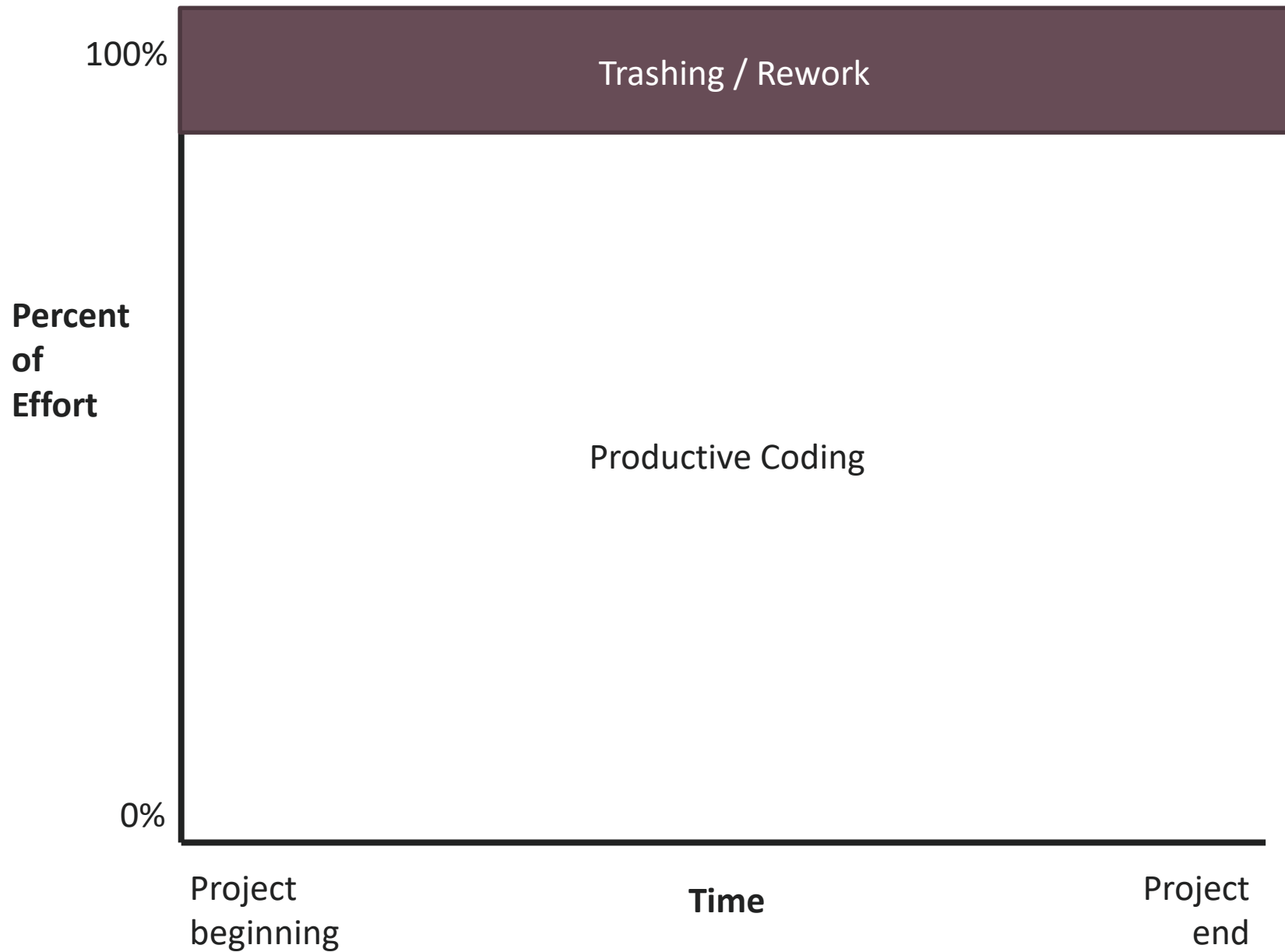
- The **spiral** software process model focuses on the construction of an increasingly-complete series of prototypes while accounting for risk.



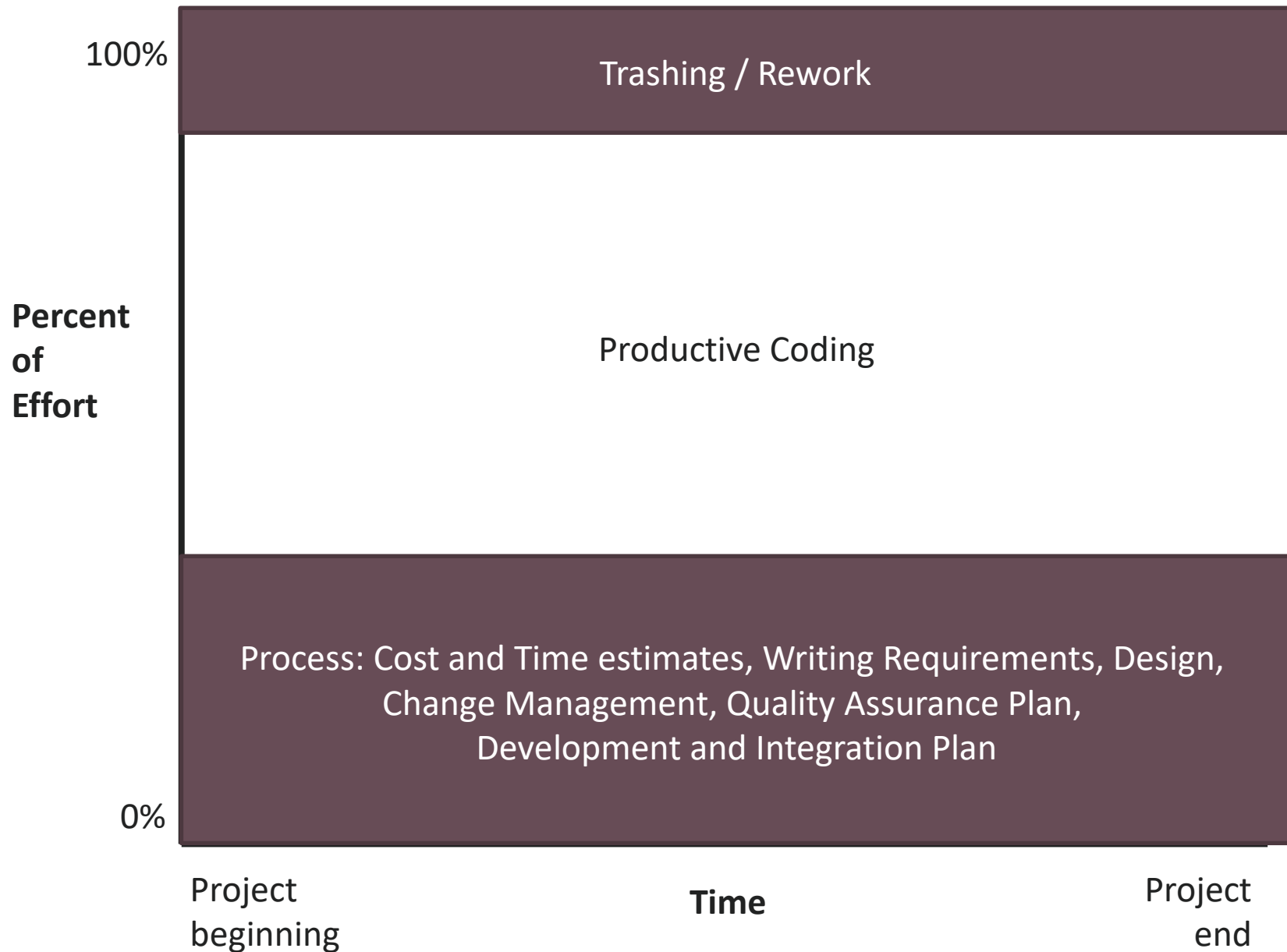
# Activity Effort over Time



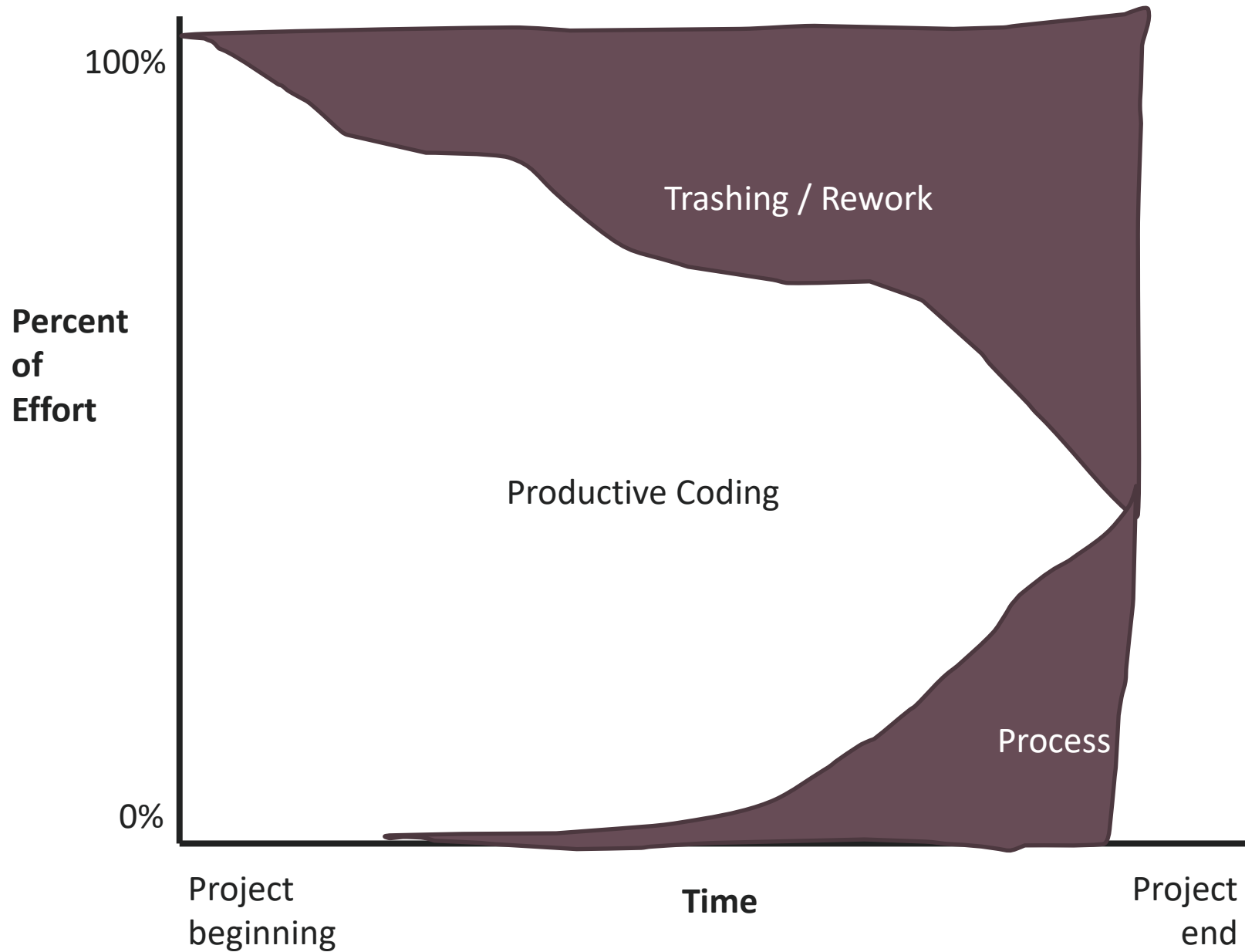
# Activity Effort over Time



# Idealized View



# Result of Failing to Plan



# Example Process Issues and Outcomes

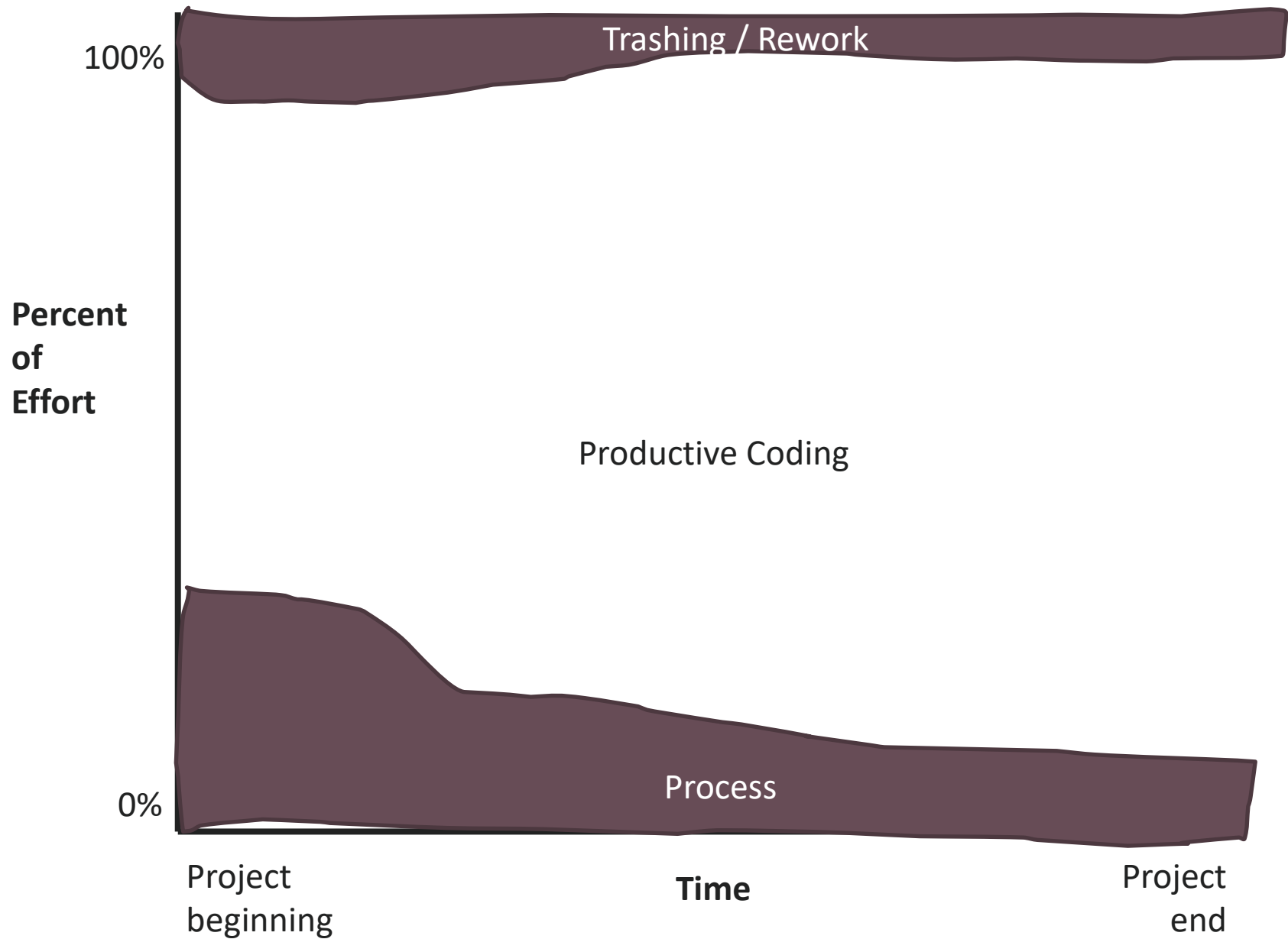
- **Requirements:** Mid-project informal agreement to changes suggested by customer or manager. → Project scope expands 25-50%
- **Quality Assurance:** Late detection of requirements and design issues. Test-debug-reimplement cycle limits development of new features. → Release with known defects
- **Defect Tracking:** Bug reports collected informally. → Bugs forgotten
- **System Integration:** Integration of independently developed components at the very end of the project. → Interfaces out of sync
- **Source Code Control:** Accidentally overwritten changes. → Lost work
- **Scheduling:** When project is behind, developers are asked weekly for new estimates. → Project falls further behind

# Survival Mode

- Missed deadlines → “solo development mode”, developers stop interacting with testers, technical writers, managers, etc.
- “The producers even set a deadline; they gave a specific date for the end of the crunch, which was still months away from the title's shipping date, so it seemed safe. That date came and went. And went, and went. When the next news came it was not about a reprieve; it was another acceleration: twelve hours six days a week, 9am to 10pm.

Weeks passed. Again the producers had given a termination date on this crunch that again they failed. Throughout this period the project remained on schedule. **The long hours started to take its toll on the team**; people grew irritable and some started to get ill. People dropped out in droves for a couple of days at a time, but then the team seemed to reach equilibrium again and they plowed ahead. The managers stopped even talking about a day when the hours would go back to normal.” - *EA: The Human Story*

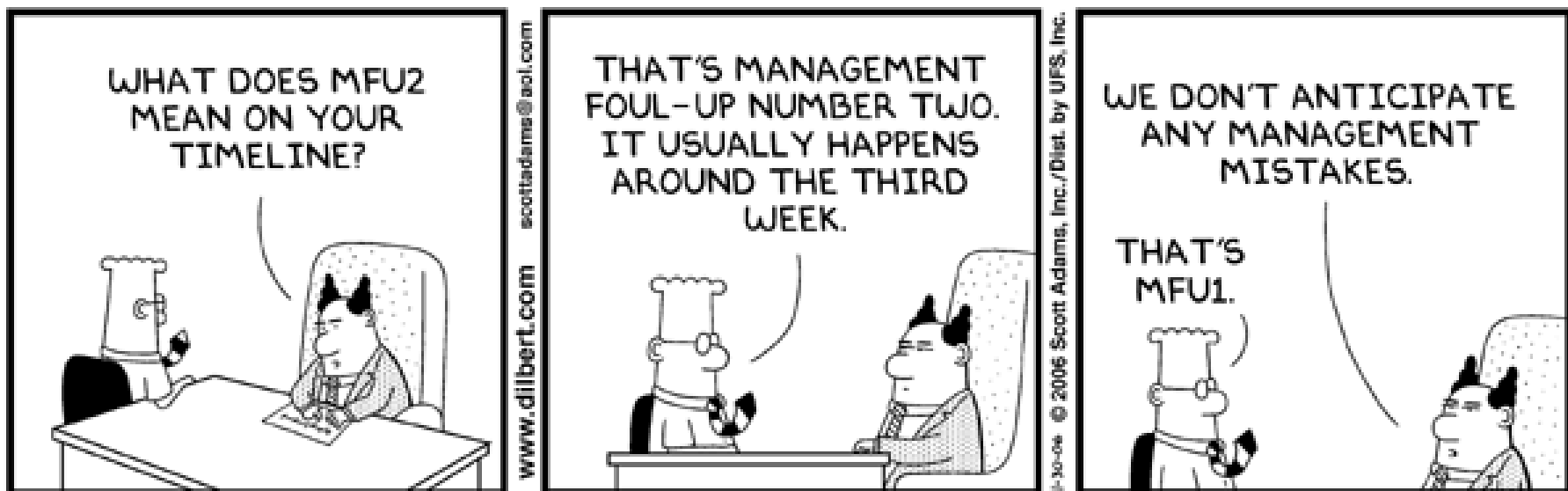
# Desired Allocation



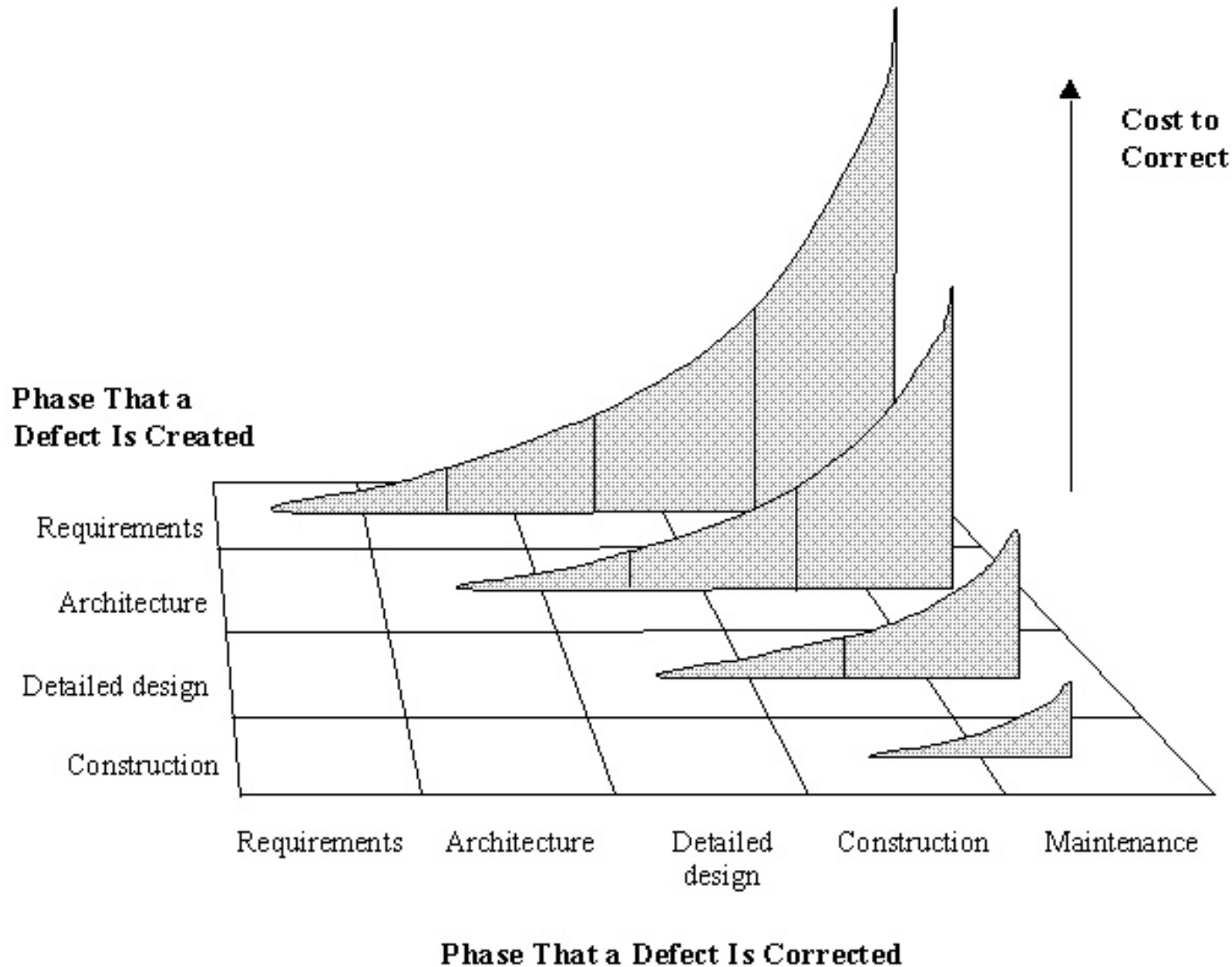


# Process Hypothesis

- A process **can increase flexibility and efficiency** for software development.
- If this is true, an up-front investment (of resources, e.g., “time”) in process can yield greater returns later on.



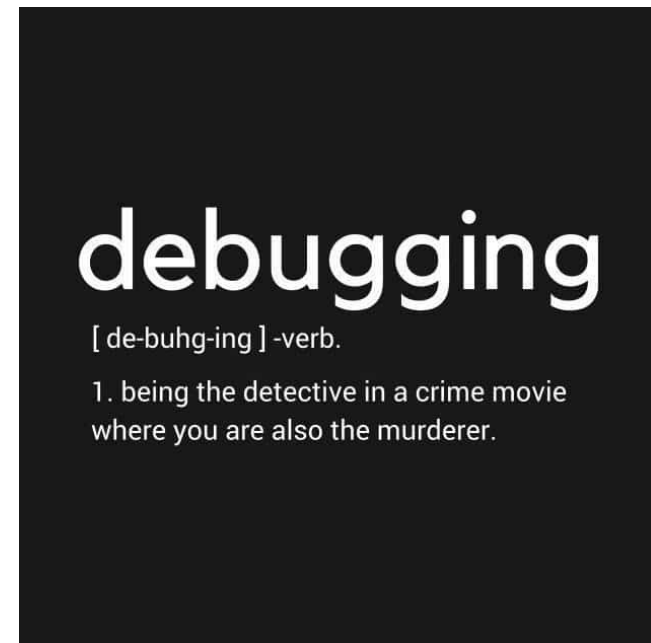
# Efficiency: Defect Cost vs. Creation Time



# Efficiency:

## Defect Cost vs. Detection Time

- An IBM report gives an average defect repair cost of
  - \$25 during coding
  - \$100 at build time
  - \$450 during testing/QA
  - \$16,000 post-release



[L. Williamson. *IBM Rational software analyzer: Beyond source code.* 2008.]

# Processes Can Be Complicated

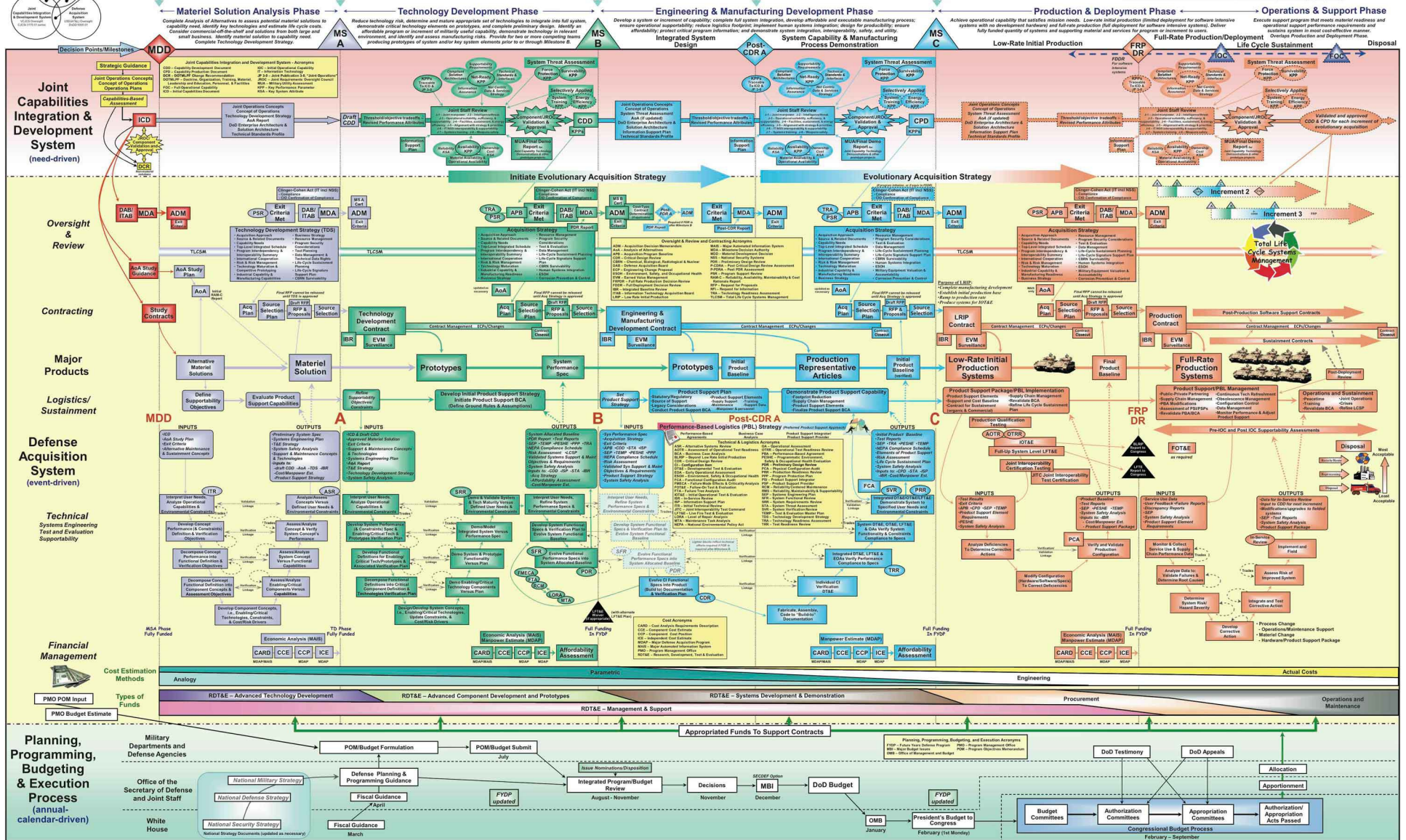
Version 5.4 15 June 2010

## Integrated Defense Acquisition, Technology, and Logistics Life Cycle Management System



Following the Materiel Development Decision, the Milestone Decision Authority may authorize entry into the acquisition process at any point, consistent with phase-specific entrance criteria and statutory requirements

This chart is a classroom aid for Defense Acquisition University students. It provides a notional illustration of interfaces among three major acquisition support systems used to develop, produce and sustain a large and complex system for national defense. Before acquisition to a complete process information is not available. For more information, see the Defense Acquisition Program Information System (DIS) website.



Author: Chuck Cochran and Brad Brown. For a single copy of this chart, send a request to [dis@dau.edu](mailto:dis@dau.edu). Send recommendations to improve the content of this chart to [chart@dau.edu](mailto:chart@dau.edu).

# Trivia: Chemistry

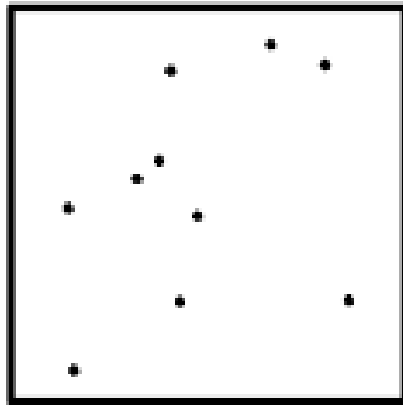
- Which of these chemical reactions would be the hardest to carry out in a school chemistry class?
  - Nitrating cellulose to produce guncotton
  - Reacting thermite with iron oxide (2500 °C)
  - Dissolving bauxite in cryolite to make aluminum
  - Cross-linking polyvinyl alcohol with sodium borate

# Trivia: Chemistry

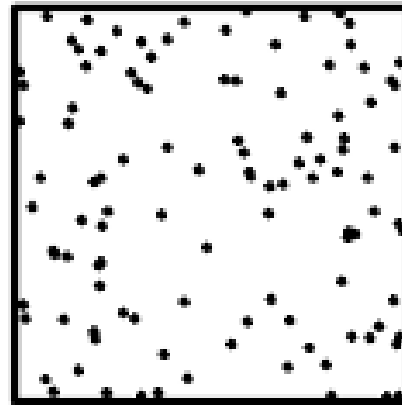
- The Hall-Héroult process (1886) extracts aluminum from the ore bauxite. Aluminum is the most abundant metallic element on Earth but not in its elemental state.
- Before this process aluminum was *more expensive than gold or platinum*:
  - “Bars of aluminum were exhibited alongside the French crown jewels at the Exposition Universelle of 1855, and Emperor Napoleon III of France was said to have reserved his few sets of aluminum dinner plates and eating utensils for his most honored guests.”

# Psychophysics

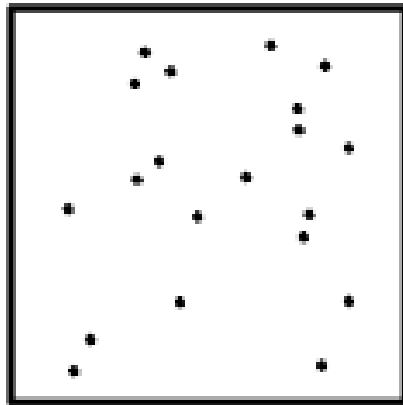
- Which two figures have the same # of dots?



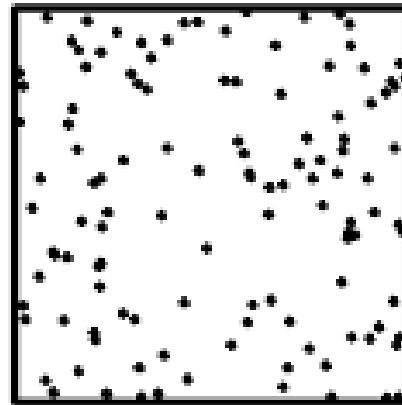
A



B



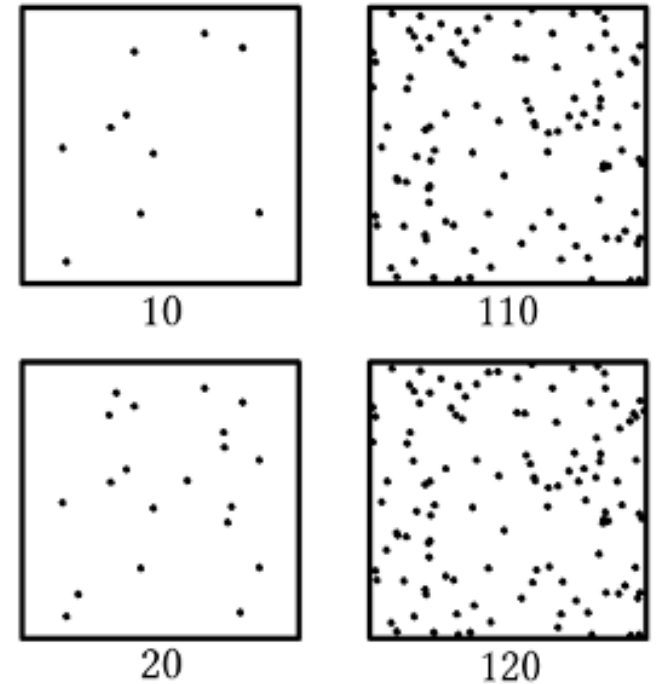
C



D

# Psychophysics: Weber's Law

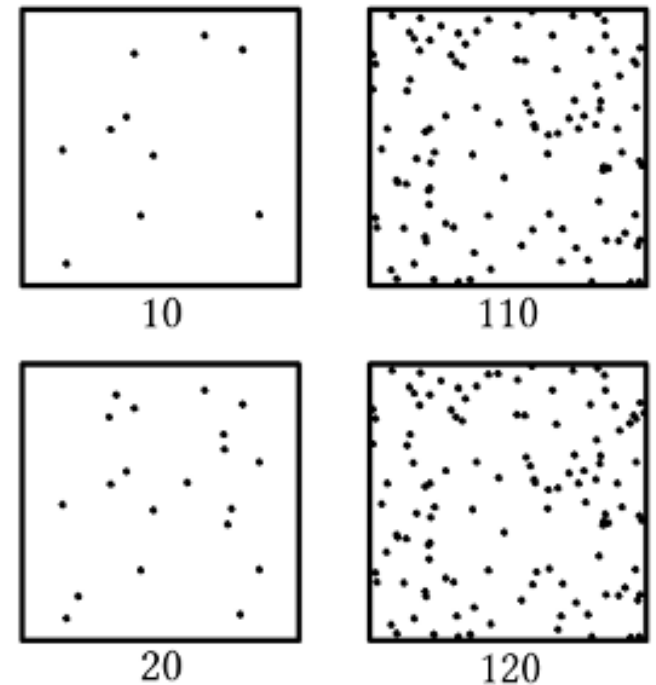
- Weber's Law states that “simple differential sensitivity is inversely proportional to the size of the components of the difference; relative differential sensitivity remains the same regardless of size.”
- That is “the perceived change in stimuli is proportional to the [size of] initial stimuli.”





# Psychophysics: Weber's Law

- That is “the perceived change in stimuli is proportional to the [size of] initial stimuli.”
- Implication for SE: Things you could notice on small-scale projects are harder to notice on large-scale projects. Your intuitions (“I can spot bugs in this”) from small class projects do *not* carry over.



# Psychology

- Consider a hypothetical cleanup scenario involving two hazardous waste sites X and Y.
  - X causes 8 cases of cancer annually (large city)
  - Y causes 4 cases of cancer annually (small city)
- Rank these three cleanup approaches:
  - A.  $X \rightarrow 4$ .  $Y \rightarrow 2$ .
  - B.  $X \rightarrow 7$ .  $Y \rightarrow 0$ .
  - C.  $X \rightarrow 3$ ,  $Y \rightarrow 3$ .

# Psychology: Zero-Risk Bias

- Three cleanup approaches:
  - A.  $X \rightarrow 4$ .  $Y \rightarrow 2$ .
  - B.  $X \rightarrow 7$ .  $Y \rightarrow 0$ .
  - C.  $X \rightarrow 3$ ,  $Y \rightarrow 3$ .
- “The bias was defined as not ranking the complete-reduction option [B] as the worst of the three options. (It should be ranked worst because it saves fewer cancer cases.) 42% of the subjects exhibited this **‘zero-risk’** bias.”

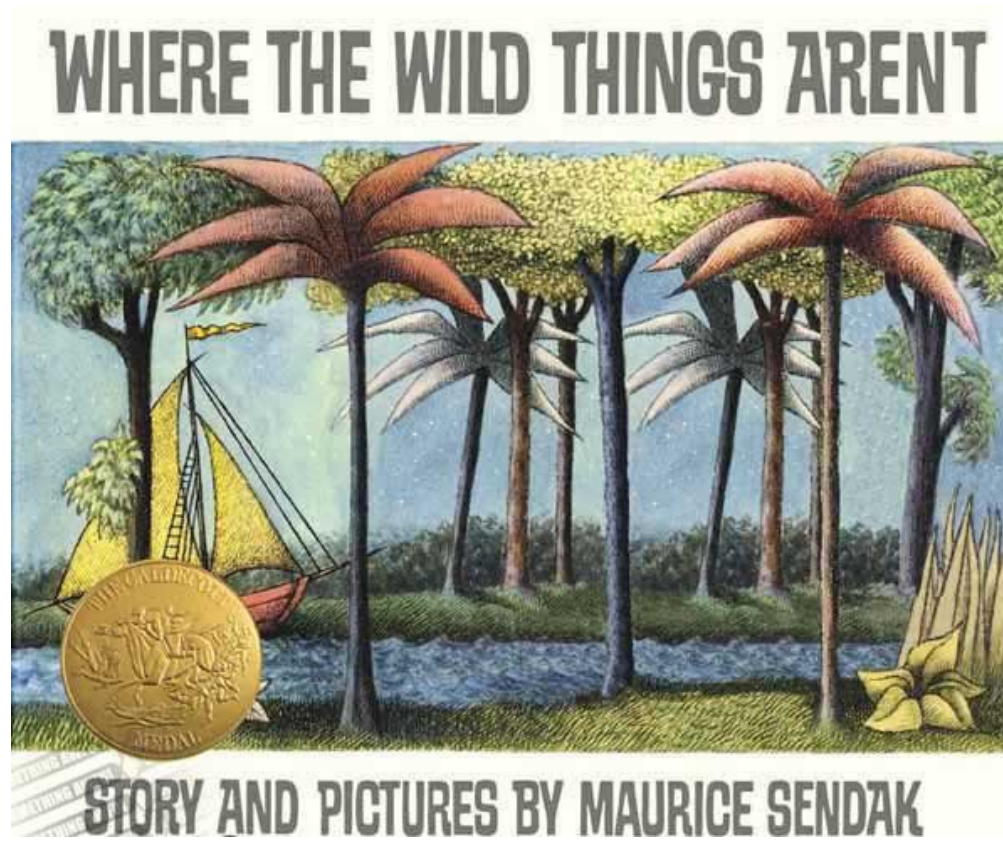
[Baron; Gowda; Kunreuther (1993). "Attitudes toward managing hazardous waste: What should be cleaned up and who should pay for it?". Risk Analysis. 13: 183-192.]

# Psychology: Zero-Risk Bias

- **Zero-risk bias** is a tendency to prefer the complete elimination of a risk even when alternative options produce a greater reduction in risk (overall).
- “42% of the subjects exhibited this **zero-risk** bias.”
  - Who? 60 CEOs of Oil and Chem Companies, 57 Economists, 94 Environmentalists, 29 Experts on Hazardous Waste, 89 Judges, 104 Legislators.
- Implications for SE: Your managers (and you) are likely to mistakenly favor risk-reduction strategies that reduce a risk to zero, even to the overall detriment of the company/product.

# Process Topics

- Estimating Effort
- Risk and Uncertainty
- Planning and Scheduling



# Estimating Time Costs

- How long would you estimate to develop a ...
  - Java Monopoly game (you alone)
  - Bank smartphone app (you with a team of four developers, one with iPhone experience, one with a security background)
- Estimate in eight-hour workdays (20 in a month, 220 per year)
- Approach: break down the task into ~five smaller tasks and estimate them. Repeat.

# Basic Plan: Learn from Experience



**EXPERIENCE**

It's what lets you recognize a mistake when you make it again.

# Constructive Cost Model

- A **constructive cost model (cocomo)** is a predictive model of time costs based on project history.
- This requires experience with similar projects.
- This rewards documentation of experience.
- Basically, it's an empirically-derived set of “effort multipliers”. You multiply the time cost by some numbers from a chart:



Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
<b>Product attributes</b>						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
<b>Hardware attributes</b>						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
<b>Personnel attributes</b>						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
<b>Project attributes</b>						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

# Can Companies Estimate?

- Study in which 35 companies bid to produce a web information system. Fourteen submitted a schedule and four were contracted to build it.

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 35, NO. 3, MAY/JUNE 2009

407

## Variability and Reproducibility in Software Engineering: A Study of Four Companies that Developed the Same System

Bente C.D. Anda, Dag I.K. Sjøberg, *Member, IEEE*, and Audris Mockus, *Member, IEEE*

**Abstract**—The scientific study of a phenomenon requires it to be reproducible. Mature engineering industries are recognized by projects and products that are, to some extent, reproducible. Yet, reproducibility in software engineering (SE) has not been investigated thoroughly, despite the fact that lack of reproducibility has both practical and scientific consequences. We report a longitudinal multiple-case study of variations and reproducibility in software development, from bidding to deployment, on the basis of the same requirement specification. In a call for tender to 81 companies, 35 responded. Four of them developed the system independently. The *firm price*, *planned schedule*, and *planned development process* had, respectively, “low,” “low,” and “medium” reproducibilities. The contractor’s *costs*, *actual lead time*, and *schedule overrun* of the projects had, respectively, “medium,” “high,” and “low” reproducibilities. The quality dimensions of the delivered products, *reliability*, *usability*, and *maintainability* had, respectively, “low,” “high,” and “low” reproducibilities. Moreover, variability for predictable reasons is also included in the notion of reproducibility. We found that the observed outcome of the four development projects matched our expectations, which were formulated partially on the basis of SE folklore. Nevertheless, achieving more reproducibility in SE remains a great challenge for SE research, education, and industry.

**Index Terms**—Software engineering life cycle, software quality, software project success, software process, multiple-case study.

What patterns  
can you spot?

Company	Firm price without VAT (Euro)	Time schedule (days)	A&D in bids	Planned effort on A&D (%)	Emphasis on A&D
1	2630	14	Brief (2)		
2	4380		Brief (2)		
3	4880		Very brief (1)		
4	4970	28	Brief (2)	30	5.0
5	8750	18	Detailed (3)	7	3.7
6	9940		None (0)	40	4.0
7	11810		Brief (2)	0	2.0
8	11880	94	Detailed (3)	26	5.6
9	12190	77	Very detailed (4)	5	4.5
10	16630		Brief (2)	12	3.2
11	18130		Very brief (1)		
12	18510	91	Brief (2)	20	4.0
13	20000	30	Detailed (3)	28	5.8
14	20020		Very brief (1)	50	6.0
15	21090		Very brief (1)	44	5.4
16	25310		Very detailed (4)	11	5.1
17	33250	49	Detailed (3)	26	5.6
18	25810		Very brief (1)		
19	25940		Brief (2)	20	4.0
20	25980		Very detailed (4)	8	4.8
21	26880	45	Detailed (3)		
22	28700	77	Very detailed (4)	10	5.0
23	28950	42	Brief (2)	30	5.0
24	29000		Brief (2)		
25	33530		Brief (2)		
26	33880	77	Detailed (3)	10	4.0
27	33900		Detailed (3)	11	4.1
28	34500		Very brief (1)	36	4.6
29	38360	63	Detailed (3)	20	5.0
30	45380		Detailed (3)	10	4.0
31	52310		Brief (2)	27	4.7
32	56900		Detailed (3)	14	4.4
33	60750		Brief (2)	43	6.3
34	69060	49	Detailed (3)	23	5.3
35	69940		Detailed (3)	6	3.6

# Results

	Company A	Company B	Company C	Company D
<b>Nationality</b>	Norwegian	Norwegian	Norwegian	International
<b>Ownership</b>	Private	By employees	By employees	Listed on exchanges
<b>Location</b>	Oslo	Oslo	Bergen	Oslo + 20 countries
<b>Size</b>	Appr. 100	Appr. 25	Appr. 8	Appr. 13,000 worldwide
<b>Firm price</b>	€20,000	€45,380	€8,750	€56,000
<b>Agreed time schedule</b>	55 days	73 days	41 days	62 days
<b>Planned effort on A&amp;D</b>	28%	20%	7%	23%

Dimensions		Company A	Company B	Company C	Company D
<b>Project</b>	<b>Contractor-related costs</b>	90 hours	108 hours	155 hours	85 hours
	<b>Actual lead time</b>	87 days	90 days	79 days	65 days
	<b>Schedule overrun</b>	58%	23%	93%	5%
<b>Product</b>	<b>Reliability</b>	Good	Good	Poor	Fair
	<b>Usability</b>	Good	Fair	Fair	Good
	<b>Maintainability</b>	Good	Poor	Poor	Good

“We found little reproducibility in the firm price of bids, and in particular, we showed that the variation in firm price was about **three times greater than in the more mature domain of road construction**. ... due partly to the paucity of standards for describing **process** and product quality.”

# Risk and Uncertainty

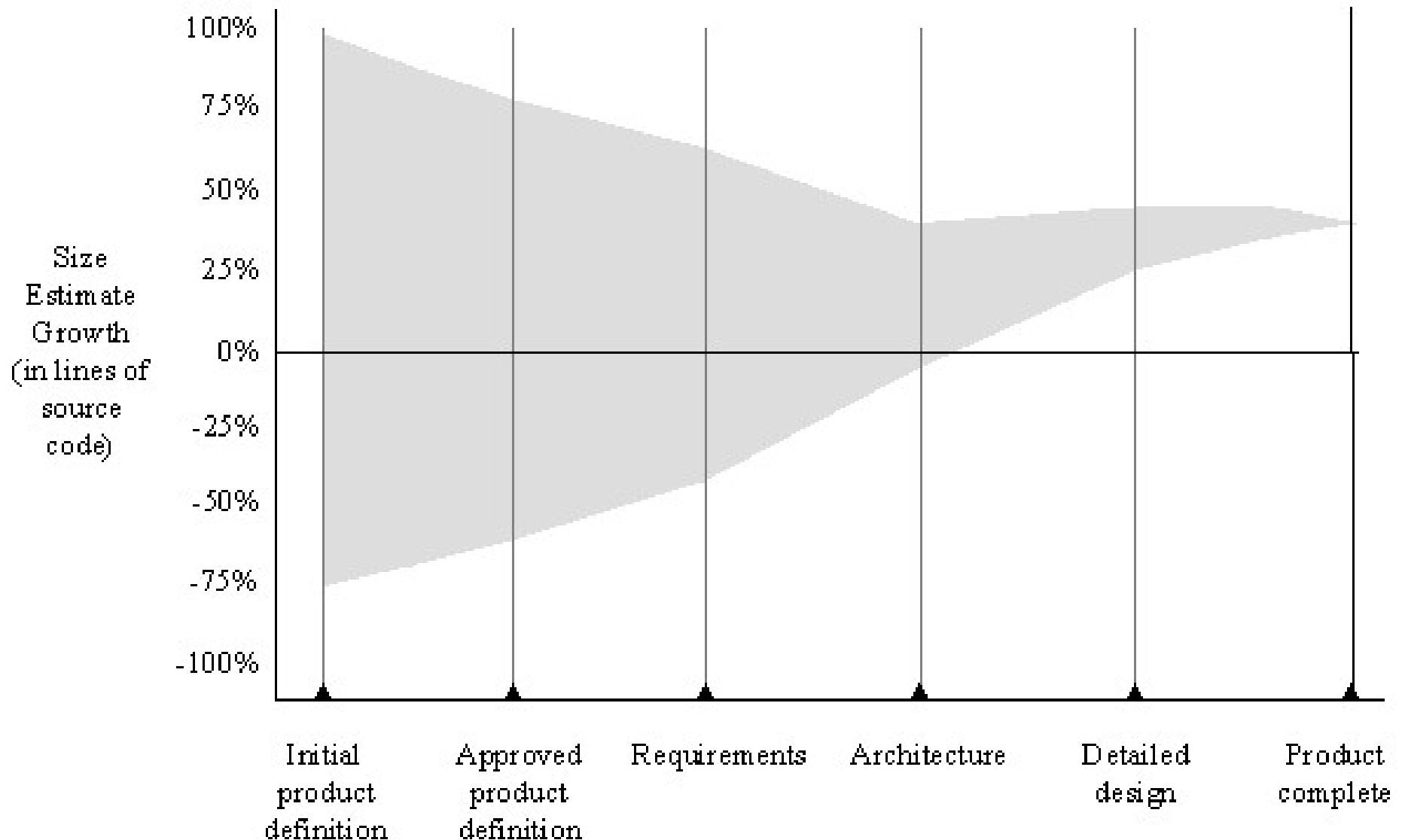


- **Risk management** is the identification, assessment, and prioritization of risks, followed by efforts to minimize, monitor and control unfortunate event outcomes and probabilities.
- Risk management is a key project management task. Examples:
  - Staff illness or turnover, product is too slow, competitor introduces a similar product, etc.

THIS ROAD MAY  
OR MAY NOT BE  
**CLOSED**  
ON OR ABOUT  
MAY 22, 2012 FOR  
APPROX 3 WEEKS



# Uncertainty Reduction Over Time



# Innovation and Risk

- Most software projects are **innovative**
  - Google, Amazon, EBay, Netflix
  - Autonomous vehicles, robotics, biomed
  - Natural language processes, graphics
- Routine projects (now, not ten years ago)
  - E-Commerce website, adaptive control systems (e.g., thermostat), 2D sprite games, etc.
- As part of the innovation cycle, routine tasks are automated ... leaving only innovative ones!



# No Catch-All Solution

- Address risk early
- Selectively innovate to increase value while minimizing risk (i.e., focus risk where needed)
- Use iteration and feedback (e.g., prototypes)
- Estimate likelihood and consequences
  - Requires experienced project leads
  - Rough estimates (e.g., <10%, <25%) are OK
  - Focus on top ten risks
- Have contingency plans

# Examples of Risk Management Strategies

Risk	Strategy
Organizational financial problems	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost-effective.
Recruitment problems	Alert customer to potential difficulties and the possibility of delays; investigate buying-in components.
Staff illness	Reorganize team so that there is more overlap of work and people therefore understand each other's jobs.
Defective components	Replace potentially defective components with bought-in components of known reliability.
Requirements changes	Derive traceability information to assess requirements change impact; maximize information hiding in the design.
Organizational restructuring	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.
Database performance	Investigate the possibility of buying a higher-performance database.
Underestimated development time	Investigate buying-in components; investigate use of a program or test generator.

# Planning

- A project should **plan** time, cost and resources adequately to estimate the work needed and to effectively **manage risk** during project execution.
- This includes scoping the work, estimating time costs, developing the schedule and budget, mitigating risks, developing quality assurance measures, etc.

Remorse pinned me against the seat for one long second. What had I just done to Jacob?

But remorse couldn't hold me very long.

**IN THAT CASE, USE A DIFFERENT WORD TO DESCRIBE THE SECOND.**

# Difficulties in Software Planning

- Typically a one-time endeavor (unique wrt. goals, constraints, organization, etc.)
- Typically involves an innovative technology
- **Intangible** results (intermediate or final) mean progress may be hard to measure
- Software projects tend to **fail more often** than other industrial projects
- (See the structured activity for a way to practice this and get a head start on HW6!)

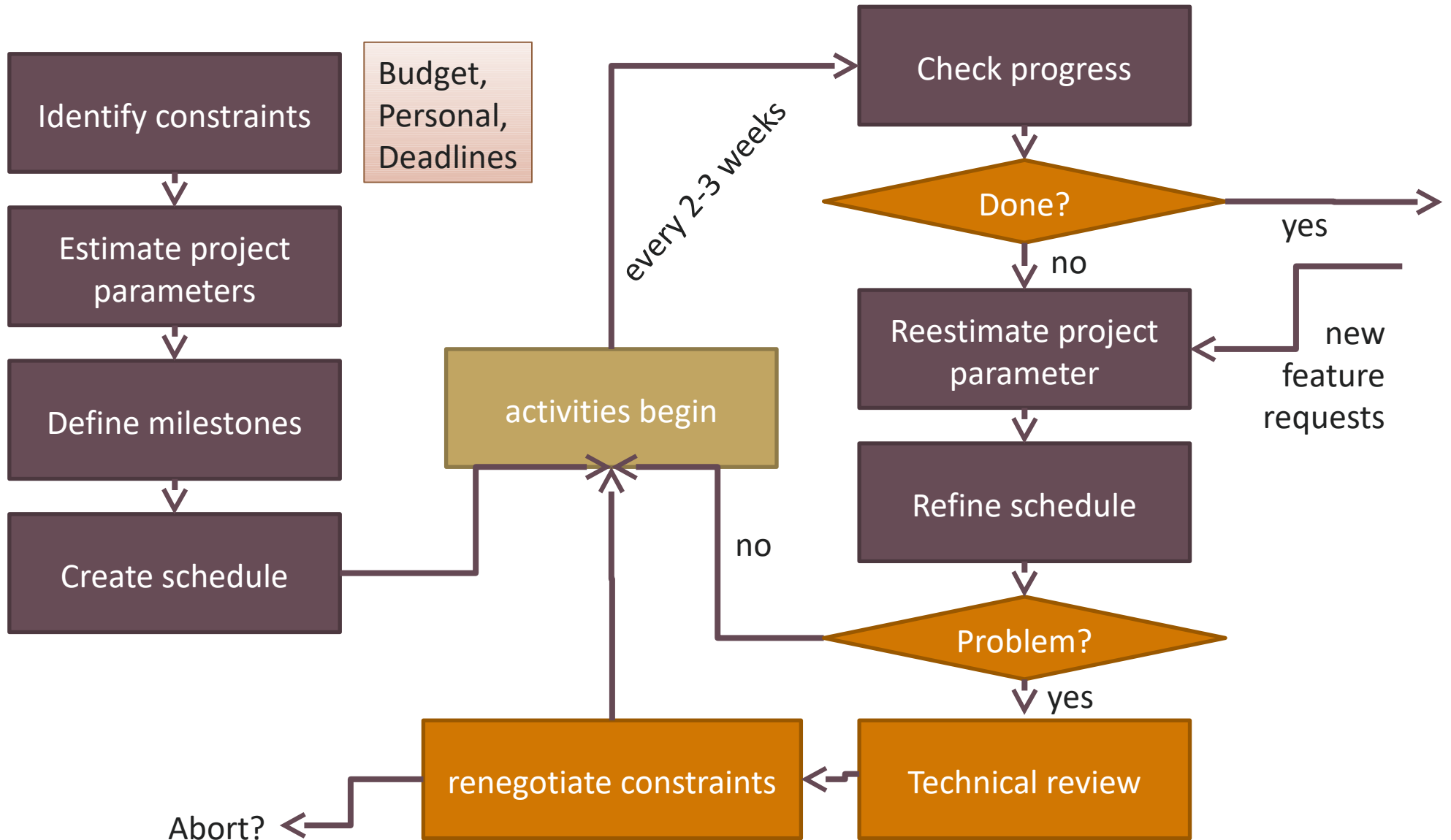
# Measuring Progress?

“I’m almost done with the app. The frontend is almost fully implemented. The backend is fully finished except for the one stupid bug that keeps crashing the server. I only need to find the one stupid bug, but that can probably be done in an afternoon. We should be ready to release next week.”

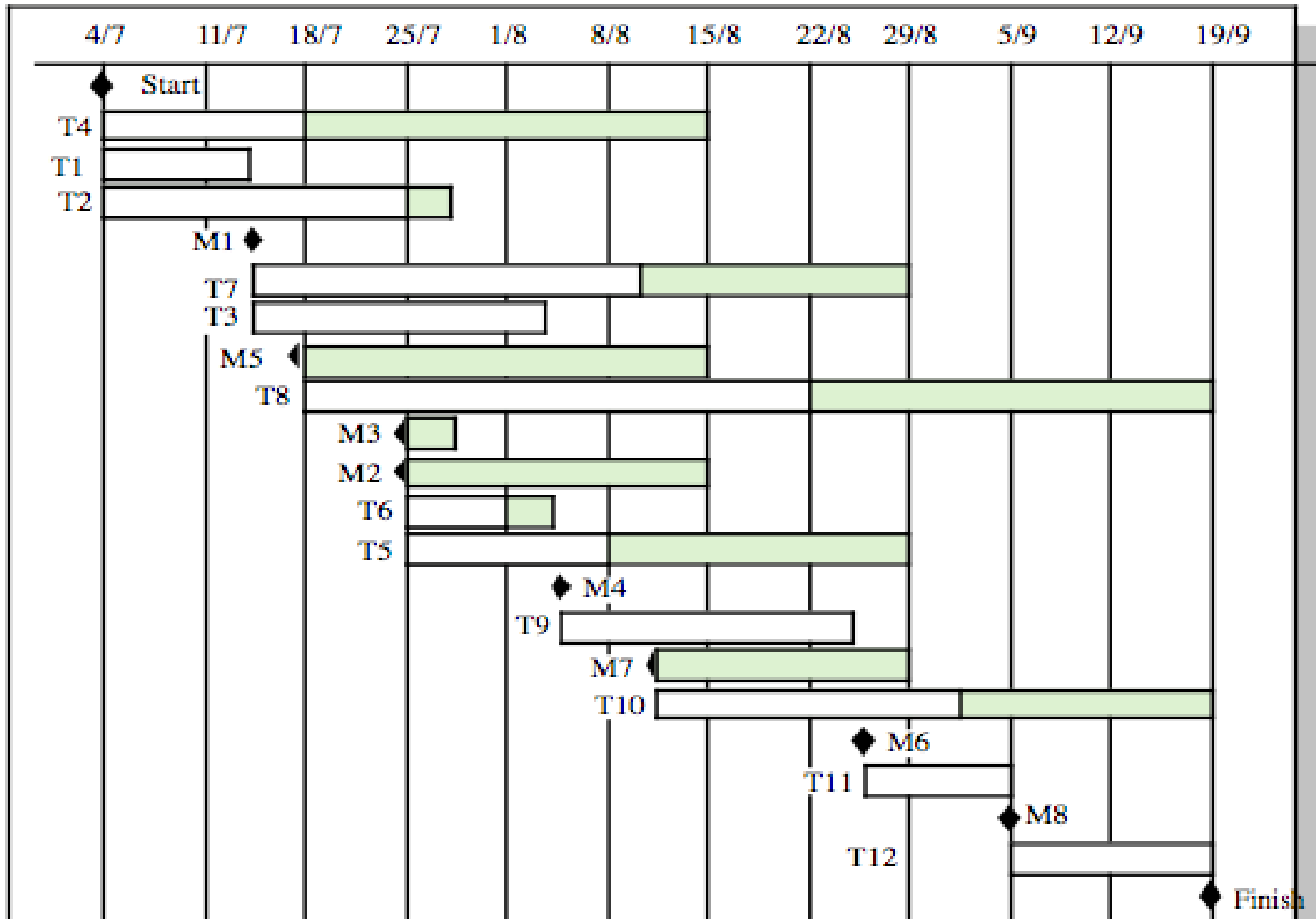
# Milestones and Deliverables

- Milestones and deliverables make intermediate progress observable, especially for software
- A **milestone** is a clean end point of a (sub)task
  - Used by the project manager
  - Reports, prototypes, completed subprojects, etc.
  - “80% done” is not a suitable milestone
- **Deliverables** are results for the customer
  - Used by the customer, outward facing

# Idealized Project Planning



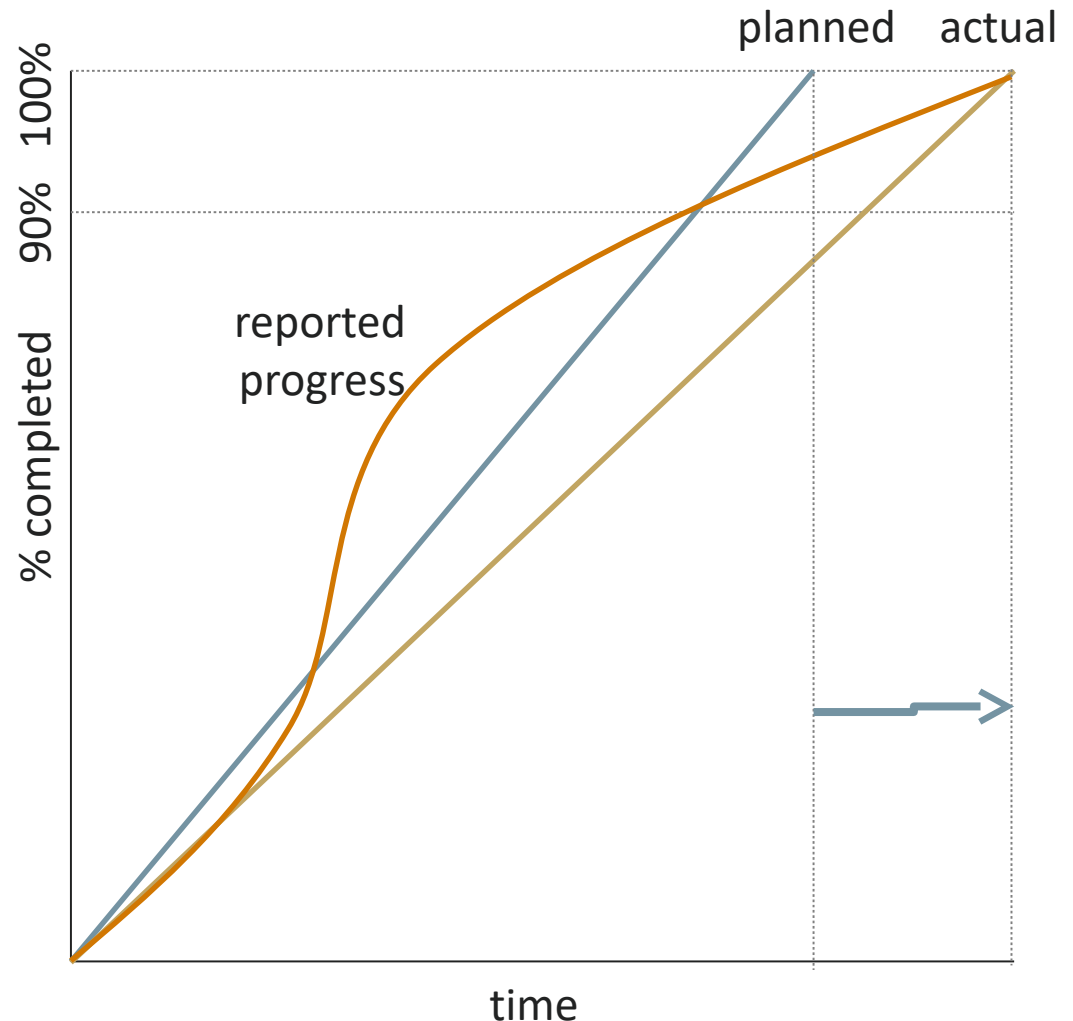
# Gantt Diagram





# Scheduling

- Inaccurate predictions are normal → update
- The “almost done” problem: the last 10% of work takes 40% of the time
  - Avoid depending entirely on developer estimates



# How does Microsoft solve this?

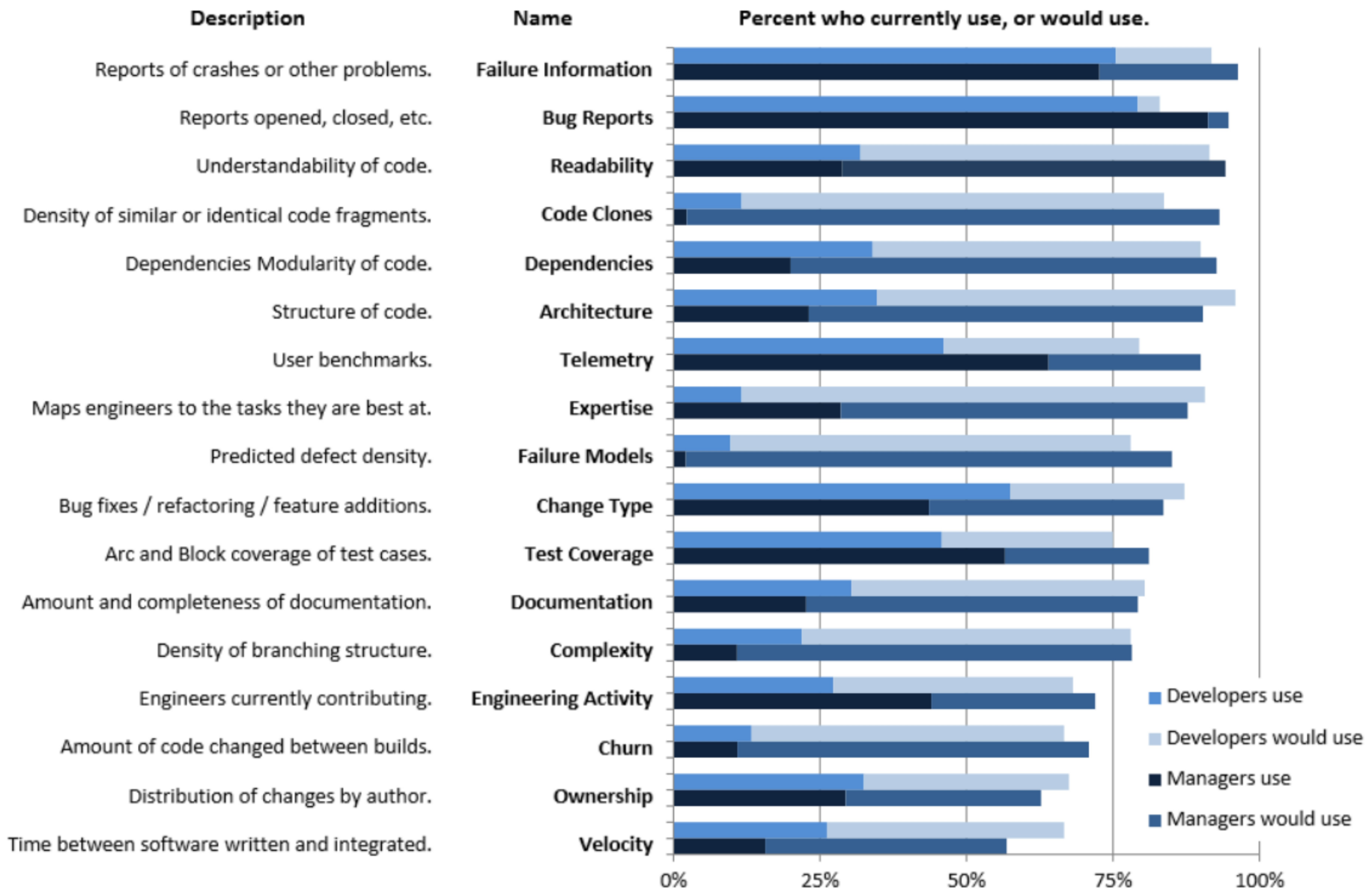


Fig. 5. Percent of managers and developers who reported that they either use or would use (if available) each of the given indicators in making decisions relevant to their engineering process.

# Story So Far

- Software processes can help, but to use them we need project planning, which needs effort estimation, which is complicated by uncertainty, which stems from risk and a lack of data.
- So ... we don't know anything?
- Stay tuned for next time for **measurement**, a potential solution to our problems.
- Reminder: **Quiz 0 due Friday!**
  - Other reading quizzes will be *surprise* pop quizzes