Foundations of Software Engineering

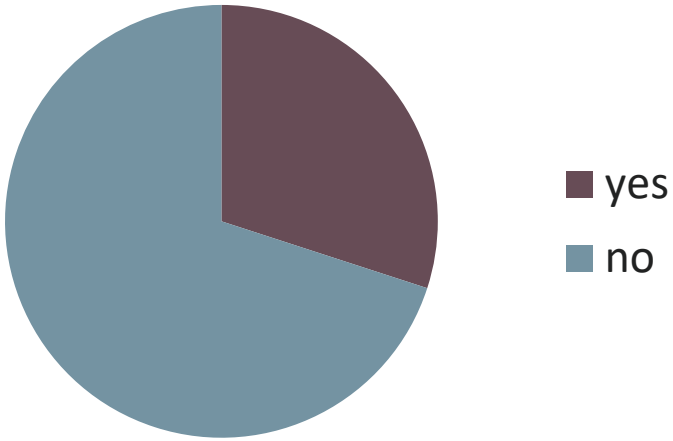
Part 2: Quick intro to process, teamwork, risk and scheduling

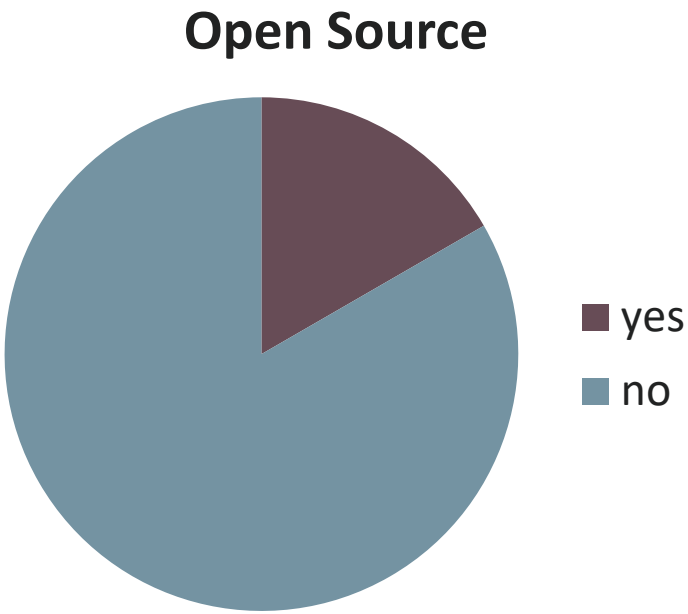
Christian Kästner

Learning Goals

- Recognize the Importance of process
- Understand the difficulty of measuring progress
- Identify what why software development has project characteristics
- Use milestones for planning and progress measurement
- Ability to divide work and planning and replan it
- Model dependencies and schedule work with network plans and Gantt diagrams
- Identifying and managing risks

About You


Saw project fail, know why
yes
no


Open Source
yes
no


Frustrating team experience
multiple times
no
once

Process

How to develop software?

1. Discuss the software that needs to be written
2. Write some code
3. Test the code to identify the defects
4. Debug to find causes of defects
5. Fix the defects
6. If not done, return to step 1

What is a Software Process?

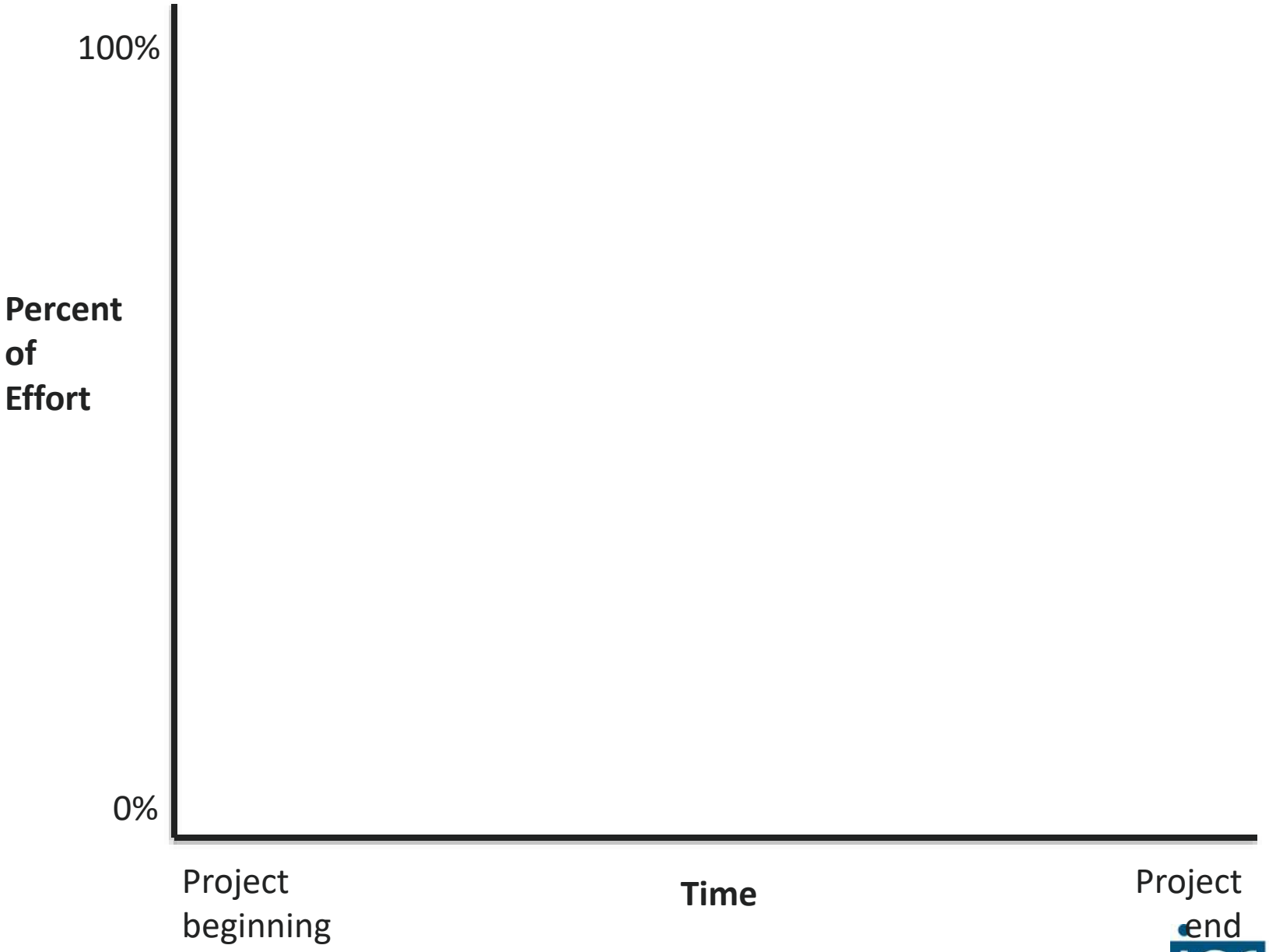
Software Process

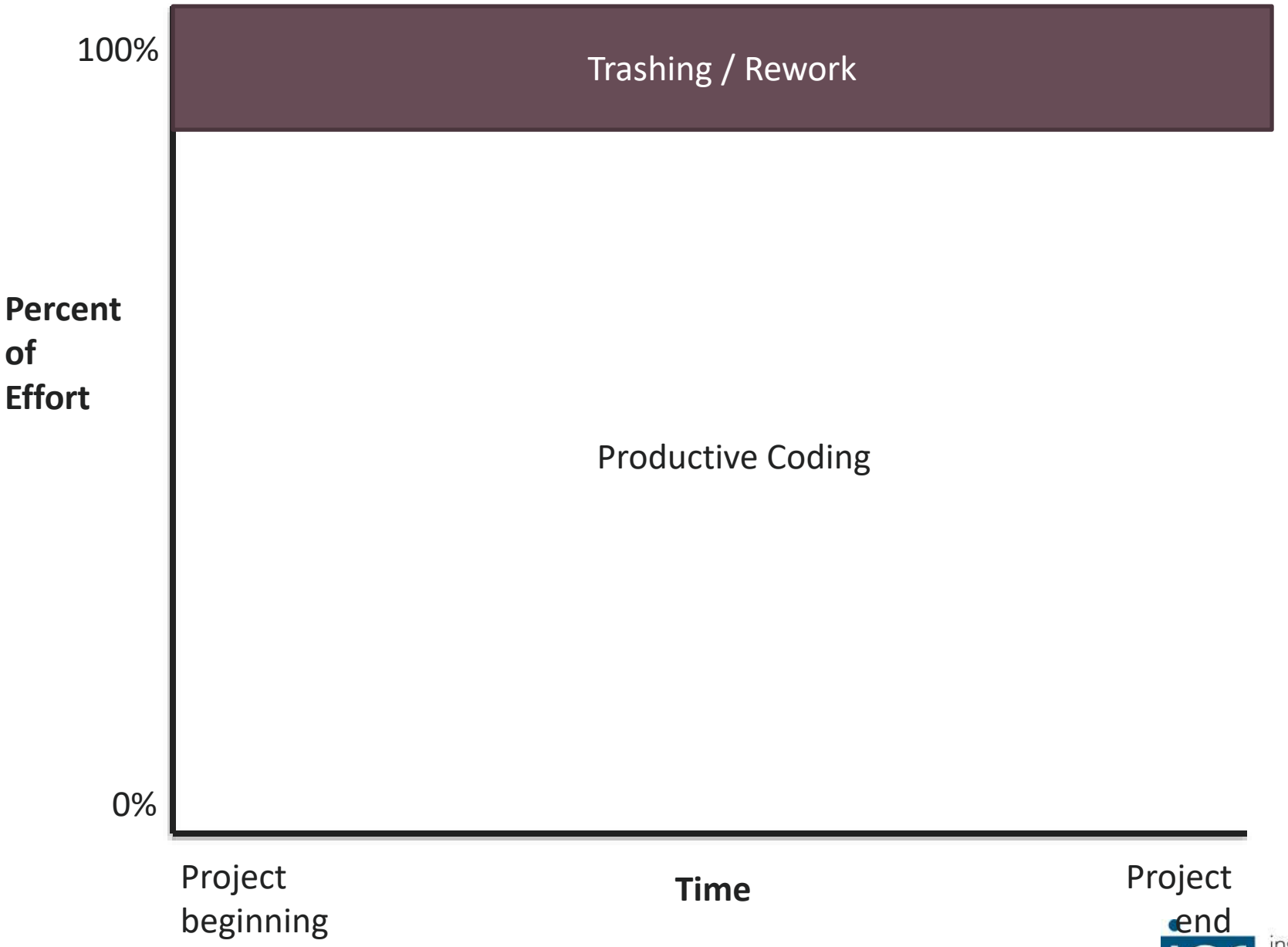
"The set of activities and associated results that produce a software product"

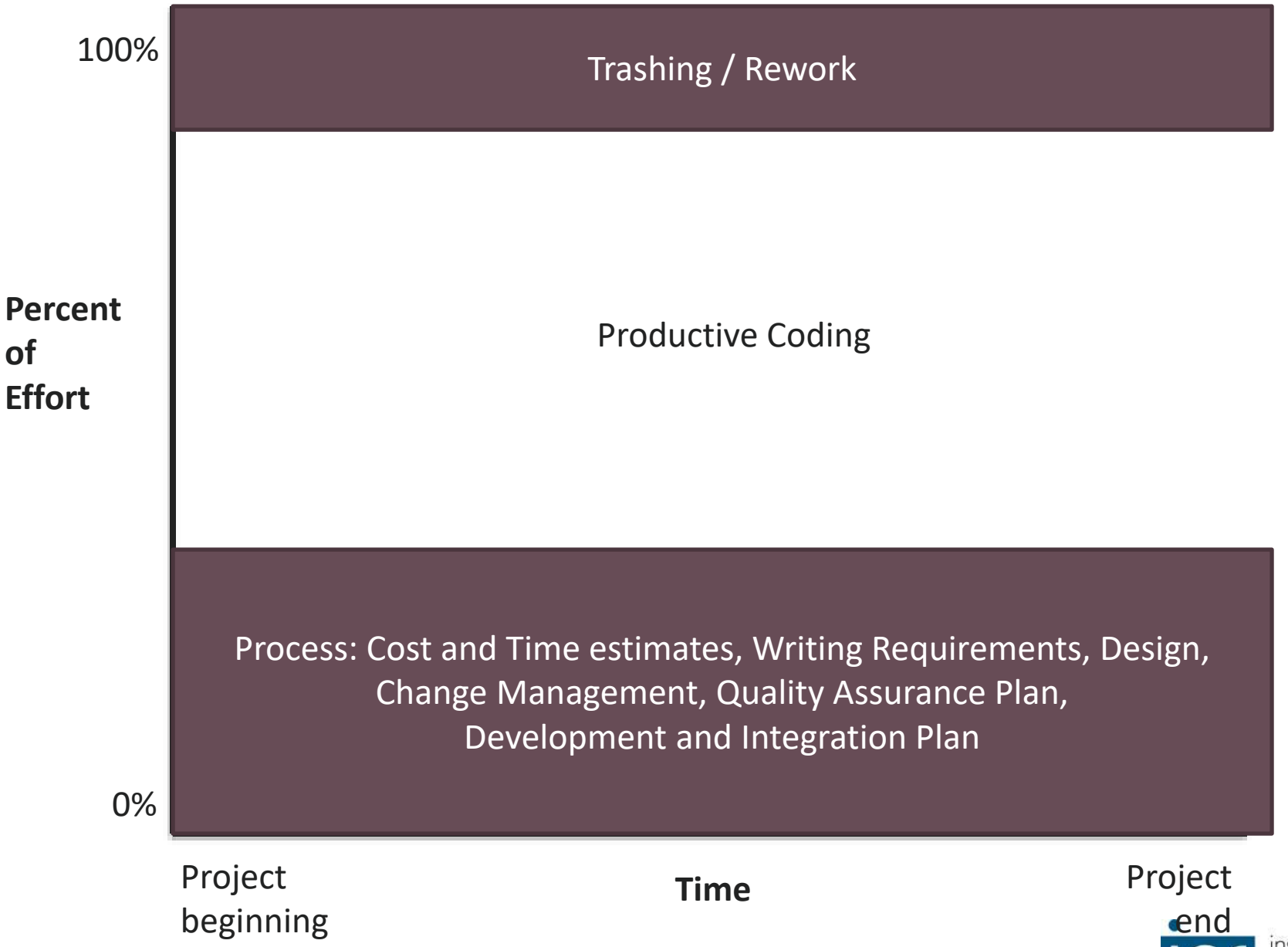
Sommerville, SE, ed. 8

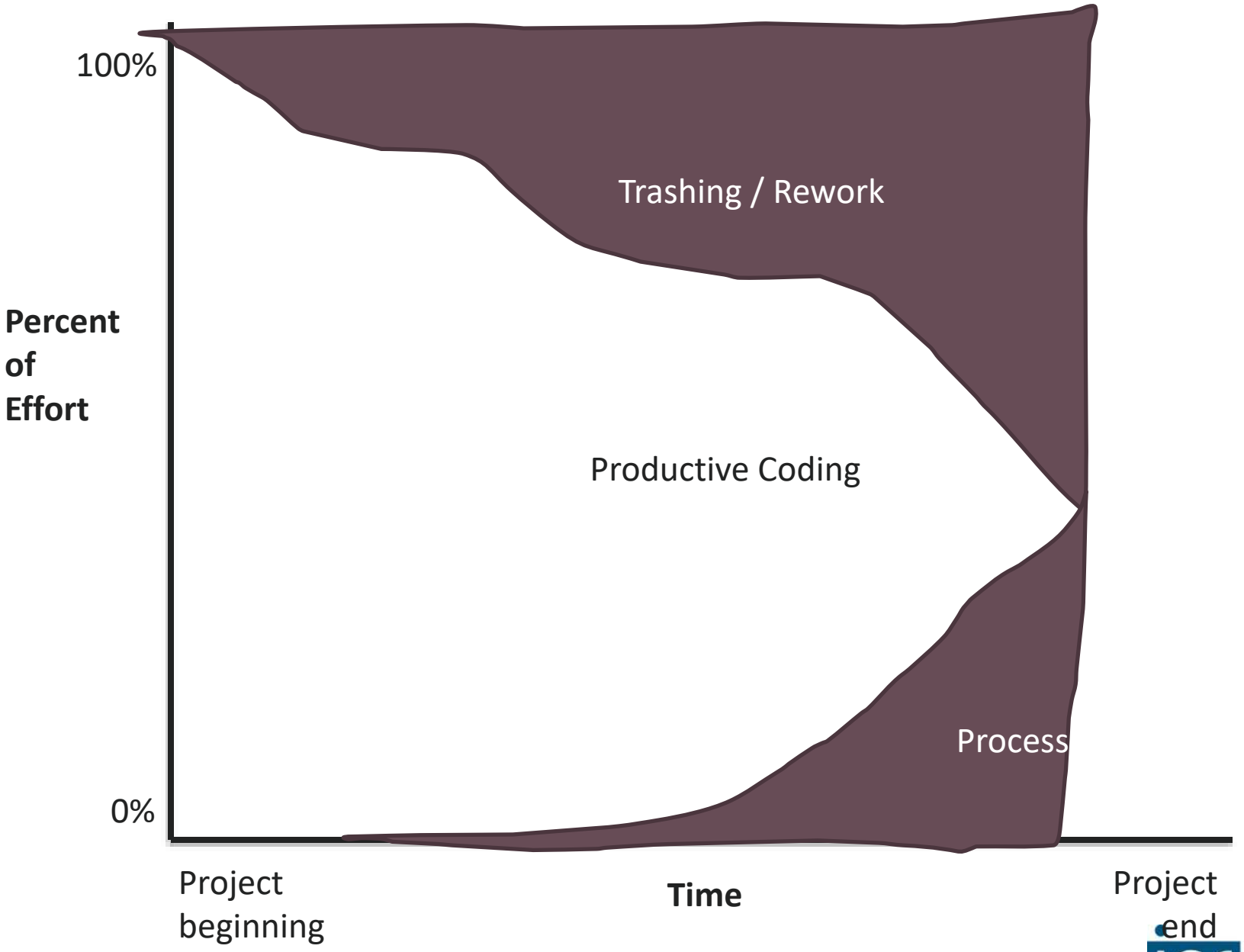
Example of Process Decisions

- Writing down all requirements
- Require approval for all changes to requirements
- Use version control for all changes
- Track all reported bugs
- Review requirements and code
- Break down development into smaller tasks and schedule and monitor them
- Planning and conducting quality assurance
- Have daily status meetings
- Use Docker containers to push code between developers and operation


100%
Percent of Effort
0%
Project beginning
Time
Project end





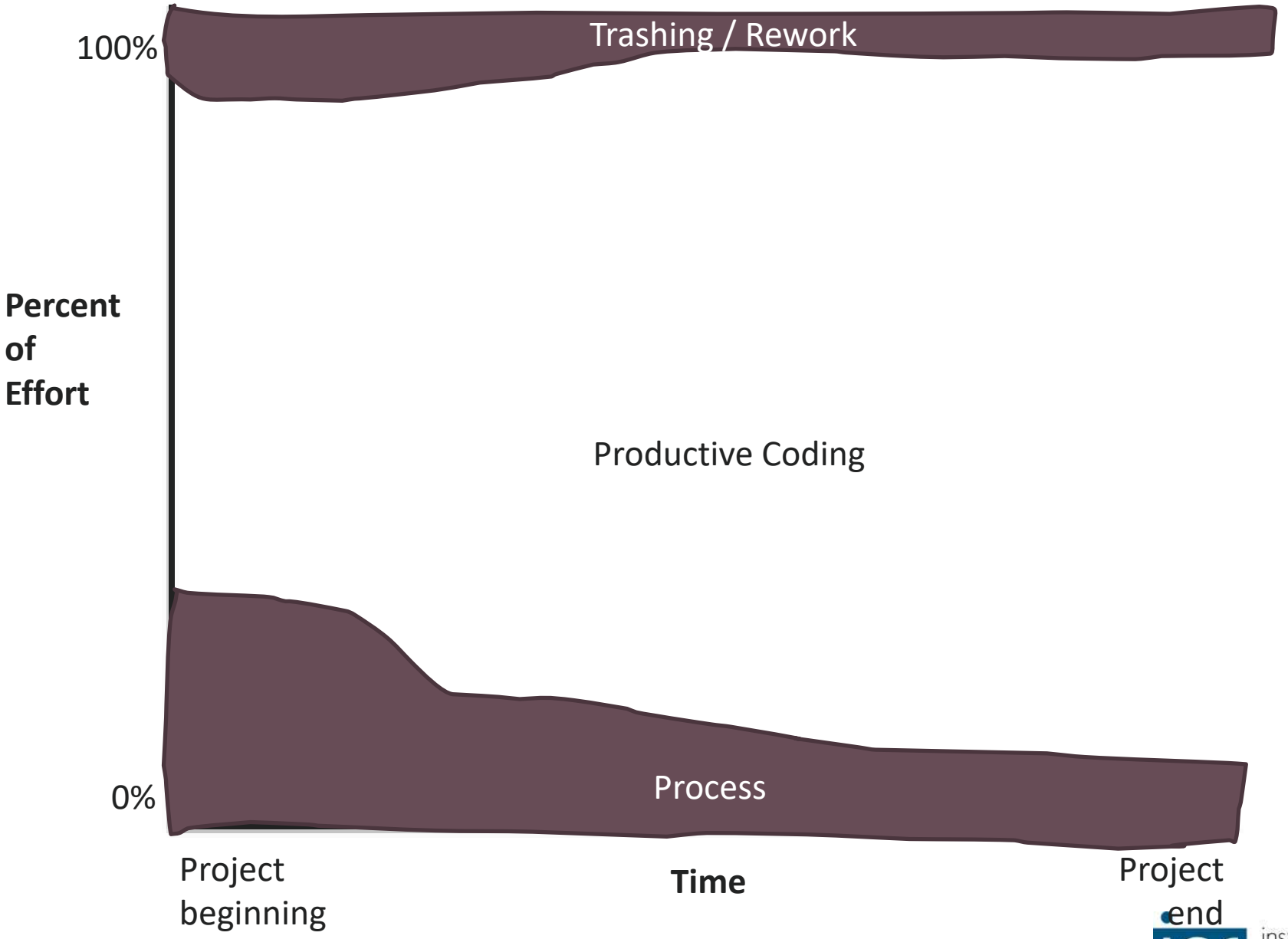


Example process issues

- Change Control: 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, 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.

Survival Mode

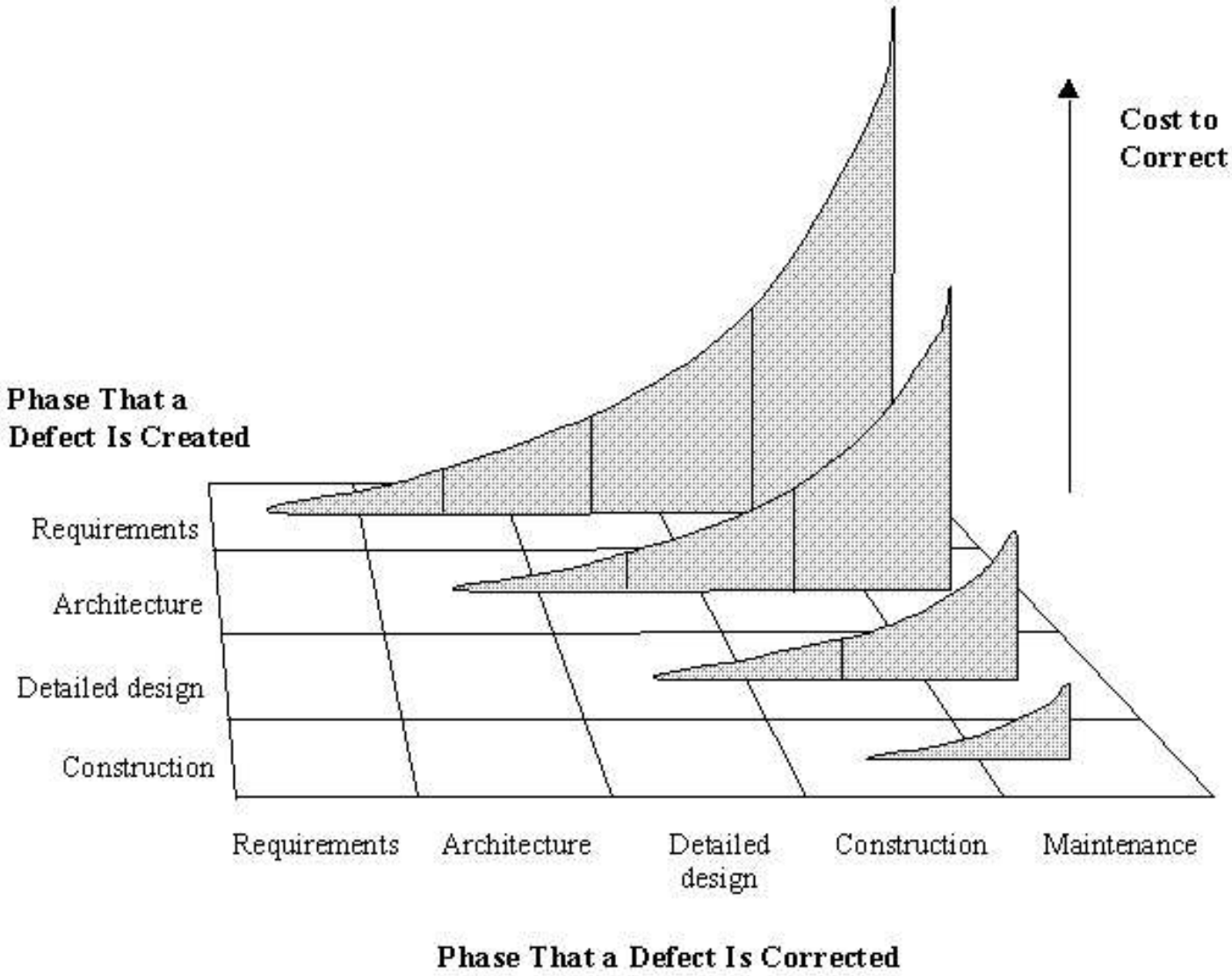
- Missed deadlines -> "solo development mode" to meet own deadlines
- Ignore integration work
- Stop interacting with testers, technical writers, managers, …



Hypothesis

- Process increases flexibility and efficiency

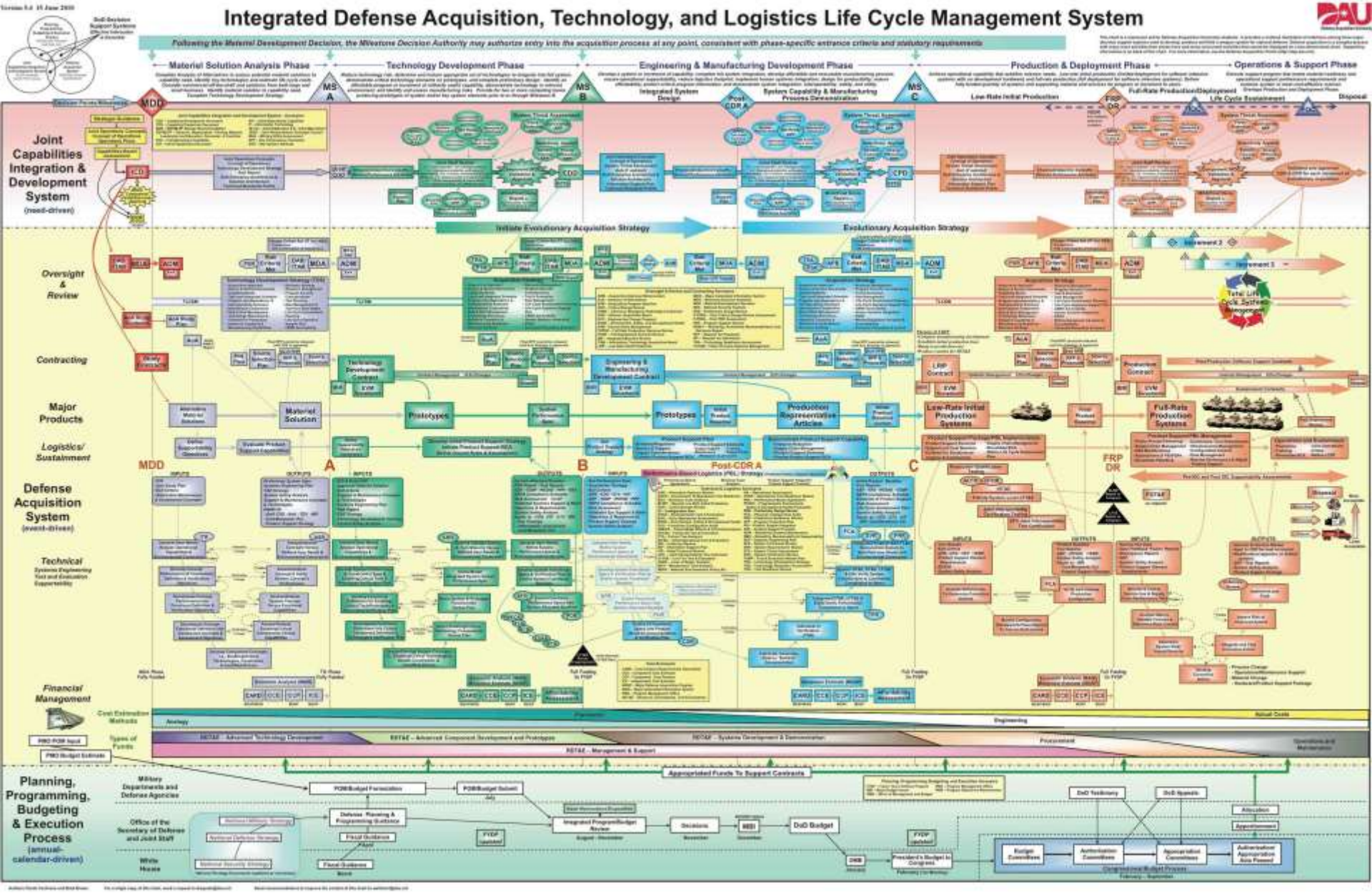
- Upfront investment for later greater returns



Process models

- Ad-hoc
- Waterfall
- Spiral
- Agile
- …

- More later



Estimating Effort

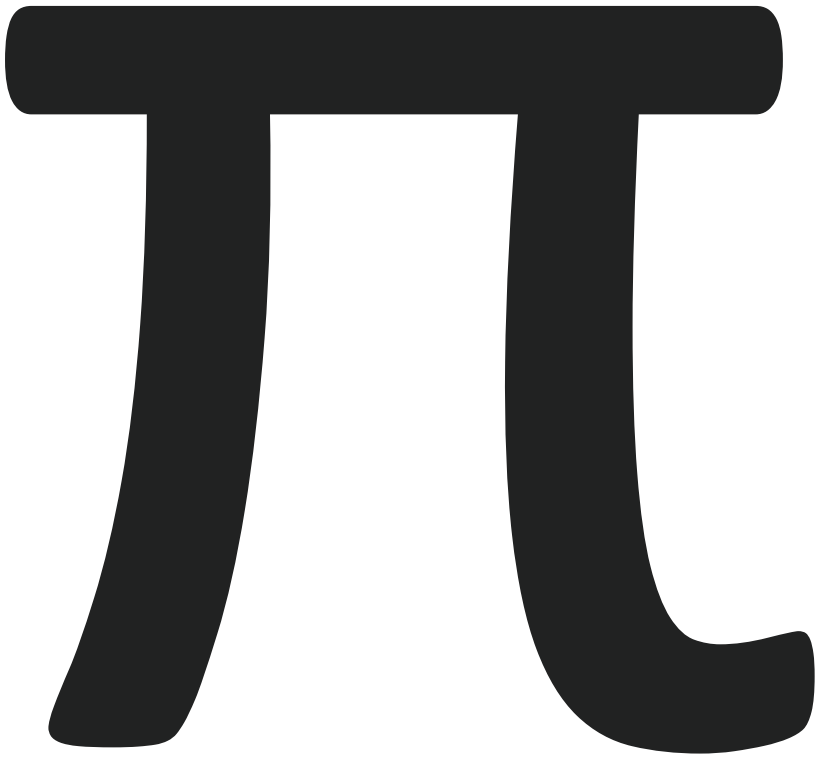
Task: Estimate Time

- A: Simple Java or web version of the Monopoly boardgame with Pittsburgh street names
  - (you)
- B: Bank smartphone app
  - (you with team of 4 developers, one experienced with iPhone apps, one with background in security)

- Estimate in 8h days (20 work days in a month, 220 per year)

Revise Time Estimate

- C: Remember Scrabble/Carcasonne experience? Is Monopoly similar/different/easier/more challenging/reusable? How much design did you do? Break down the task into ~5 smaller tasks and estimate them. Revise your overall estimate if necessary



Constructive Cost Model (Cocomo)

- Regression formula based on project history
- Requires experience with similar projects
- Encourages documentation of experience

Constructive Cost Model (Cocomo)

Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
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
Hardware attributes						
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	
Personnel attributes						
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		
Project attributes						
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	

http://en.wikipedia.org/wiki/Cocomo; Sommerville, SE, Chapter 26

Study: Variability and Reproducibility in Software Engineering

- Study by Simula Research Lab in Norway
- Created System Requirements Specification for a web information system (11 pages)
- Received bids from 35 companies, 14 with schedule
- Contracted 4 companies to build the same system

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

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

Bids and time estimations

- No relationship between price and planned time or methods in bids

	Dimension	No. of companies	CV	1/CV	Reproducibility
Bids	Firm price	35	0.65	1.5	Low
	Time schedule	14	0.49	2.0	Low
	Emphasis on A&D*	27	0.20	4.9	Medium
Projects	Contractor-related costs	4	0.29	3.4	Medium
	Actual lead time	4	0.14	7.1	High
	Schedule overrun	4	0.87	1.1	Low
Products	Reliability	4	0.48	2.1	Low
	Usability	4	0.17	5.9	High
	Maintainability	4	0.46	2.2	Low

Tender Prices for Six Projects of the Norwegian Public Roads Administration

Construction type	N companies	Stddev	Mean	CV	1/CV
Bridge	2	15,000	45,000	0.3	3.0
Electrical installation	6	13,000	49,000	0.3	3.7
Electrical installation	7	15,000	57,000	0.3	3.9
Road maintenance	5	4,400	47,000	0.1	10.5
Road maintenance	3	2,900	16,000	0.2	5.6
Road maintenance	3	9,800	47,000	0.2	4.8
Mean	**4**	**10,000**	**44,000**	**0.2**	**5.2**

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

Anda, Bente CD, Dag IK Sjøberg, and Audris Mockus. "Variability and reproducibility in software engineering: A study of four companies that developed the same system." *IEEE Transactions on Software Engineering* 35.3 (2009): 407-429.

Development Process

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

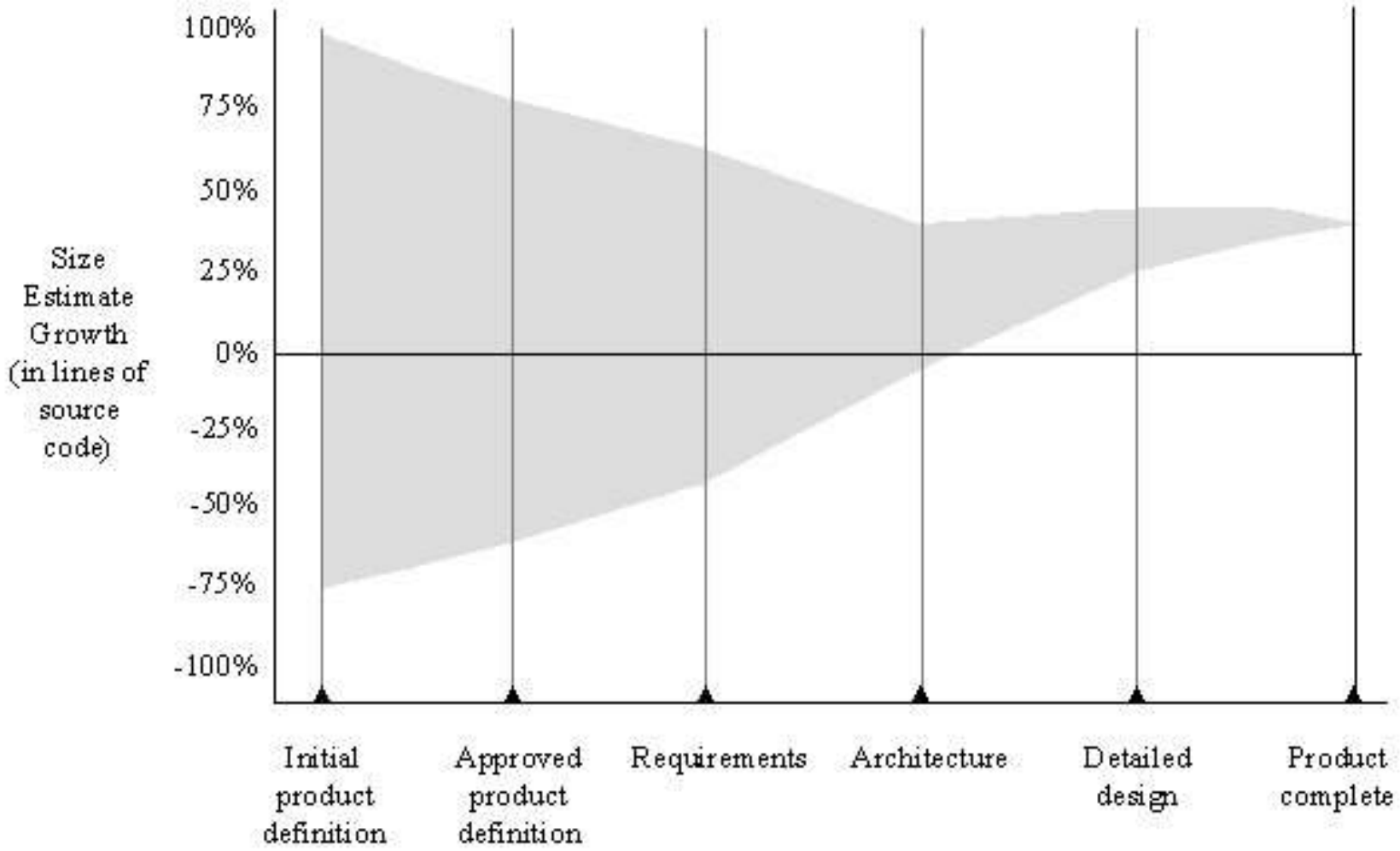
Dimensions		Company A	Company B	Company C	Company D
Project	**Contractor-related costs**	90 hours	108 hours	155 hours	85 hours
	Actual lead time	87 days	90 days	79 days	65 days
	Schedule overrun	58%	23%	93%	5%
Product	**Reliability**	Good	Good	Poor	Fair
	Usability	Good	Fair	Fair	Good
	Maintainability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

Risk and Uncertainty

Risk management

- Key task of a project manager
- Identify and evaluate risks early
- If necessary, plan mitigation strategies
- Document results of risk analysis in project plan

- Project risks: scheduling and resources
  - e.g., staff illness/turnover
- Product risks: Quality and functionality of the product
  - e.g. used component too slow
- Business risks:
  - e.g., competitor introduces similar product



Sources of Uncertainty

- Unpredictable operating environment
  - Cybersecurity threats, device drivers
  - Unanticipated usage scenarios
- Limited predictive power of models
  - Halting, abstract interpretation, testing
- Bounded rationality of humans
  - Designers, developers
  - Customers, users

Innovative vs Routine Projects

- Most software projects are innovative
  - Google, Amazon, Ebay, Netflix
  - Vehicles and robotics
  - Language processing, Graphics
- Routine (now, not 10 years ago)
  - E-commerce websites?
  - Many control systems?
  - Routine gets automated -> innovation cycle

Case 1

Case 2

NORWIK JUGGLING

Username ✱ •••••••• ✱

Home | About Us | Shop | Media | Contact us

You are here: Home › Shop

Big
75mm 67 grams
6.00 EUR
Buy Detail

Small
60mm 53 grams
4.00 EUR
Buy Detail

Viking Norwik
75mm 134g-402g [134g 7€ | [201g 8€ | [268g 9€ | [335g 10€ | [402g 11€ |
[illegible]
Buy Detail

Discussion

- What are important risks in early and late phases of development?
- Analyze risks to rank them
- How can they be managed/mitigated?

Accepting and Coping with Risks

- Selectively innovate to increase value
- Improve capability and competitiveness

- Focus risk where it is needed
- Rely on precedent and convention (experience)
- Use iteration and feedback
  - prototypes, spiral development, sprints

Managing Risks

- Address risk early
- Prototyping, spiral development
- Identify mitigation strategies

Example: Bank Software

- Build on vendor and open-source components
  - OS, DB, app server, tooling, backups, information management, …
  - Stick with reliable vendors and integrators
- Focus internal risk-taking on business differentiators
  - Web-site features, internal queries, decision algorithms
- Outsource to access expertise, lower costs, gain flexibility
  - Black-box testing
  - Overall system architecture
  - Internal engineering practices and tools

Risk analysis

- Estimate likelyhood and consequences
- Requires experienced project lead
- Rough estimations usually sufficient
  - very low (<10%), low (<25%), ...
  - catastrophic, severe, acceptable, neglegible
- Focus on top 10 risks

Risk planning

- Risk avoidance
- Risk minimization
- Emergency plans

Strategies to help manage risk

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 generator.

Planning

(Software) Projects

- One-time endeavour; unique wrt
  - Goals
  - time, financial, personal, and other constraints
  - differences to other endeavours
  - project-specific organization
- Defined, limited time (clear start and end)
- Clear goals
- Constraints: budget, resources, ...
- Requires management
  - high risk
  - coordination of experts

- Software development always proceeds in projects
- New/unique goals, innovatice technology
- Intangable result, progress hard to measure

- Software projects tend to fail more often than industrial projects

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.”

Measuring Progress?

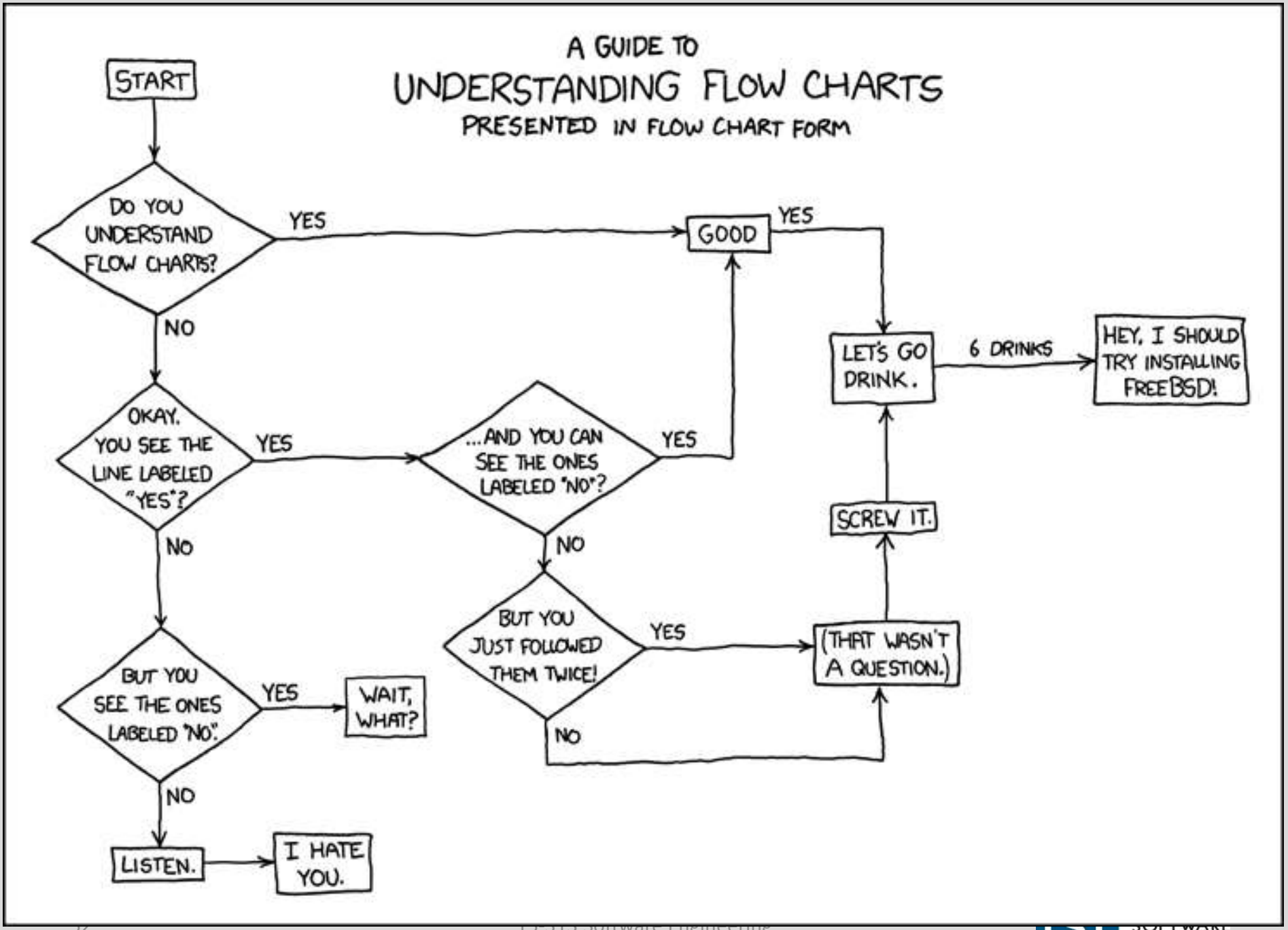
- Developer judgment: x% done
- Lines of code?
- Functionality?
- Quality?

Milestones and deliverables

- Making progress observable, especially for software
- Milestone: clear end point of a (sub)tasks
  - For project manager
  - Reports, prototypes, completed subprojects
  - "80% done" not a suitable mile stone
- Deliverable: Result for customer
  - Similar to mile stone, but for customers
  - Reports, prototypes, completed subsystems

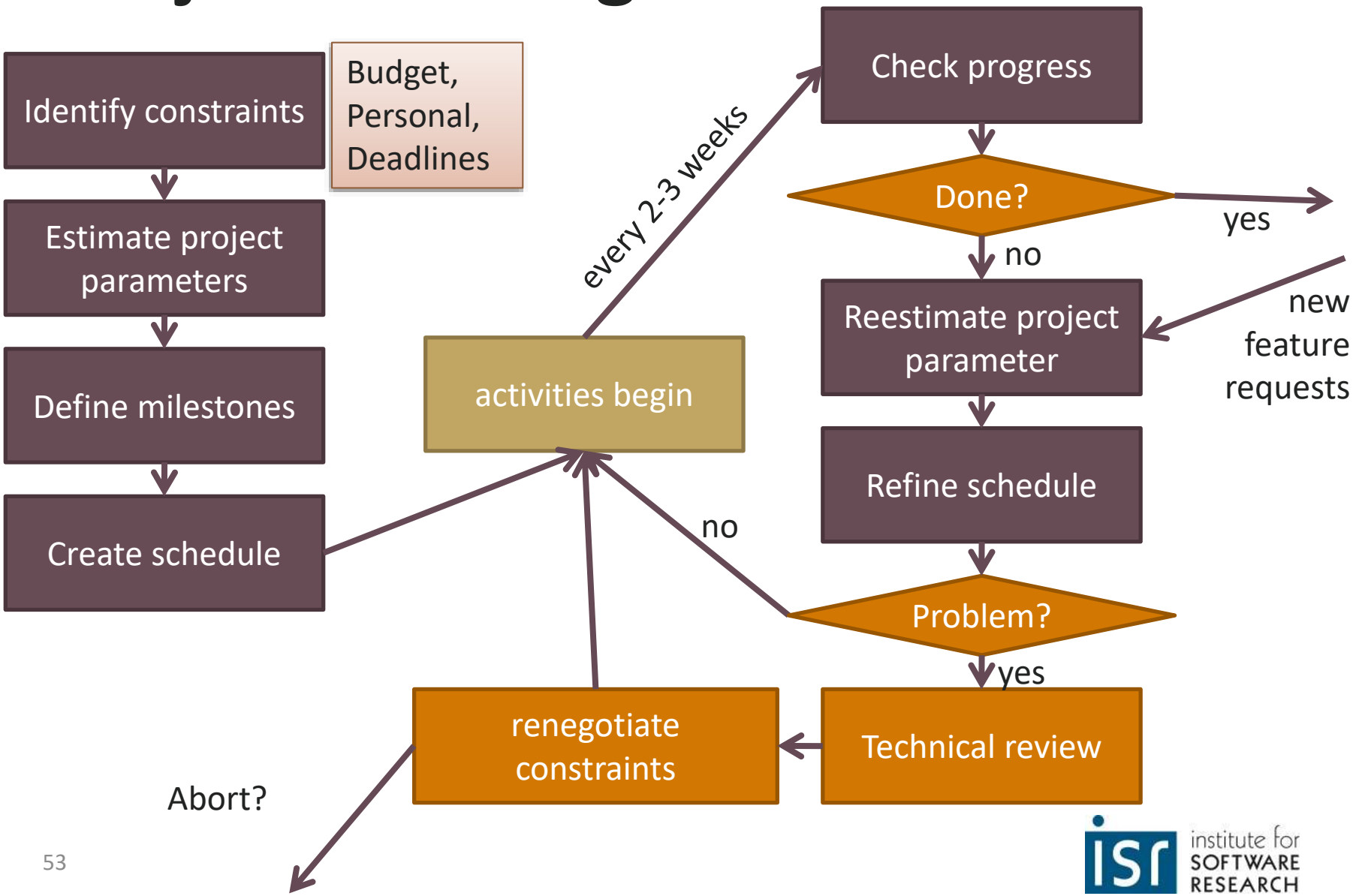
Case 3

- Define tasks and milestones for the Monopoly implementation
- Which tasks have dependencies?


A GUIDE TO
UNDERSTANDING FLOW CHARTS
PRESENTED IN FLOW CHART FORM
START
DO YOU UNDERSTAND FLOW CHARTS?
YES
GOOD
YES
NO
LET'S GO DRINK.
6 DRINKS
HEY, I SHOULD TRY INSTALLING FREEBSD!
OKAY. YOU SEE THE LINE LABELED "YES"?
YES
...AND YOU CAN SEE THE ONES LABELED "NO"?
YES
SCREW IT.
NO
NO
BUT YOU JUST FOLLOWED THEM TWICE!
YES
(THAT WASN'T A QUESTION.)
BUT YOU SEE THE ONES LABELED "NO".
YES
WAIT, WHAT?
NO
NO
LISTEN.
I HATE YOU.

xkcd.com/518

Project Planning

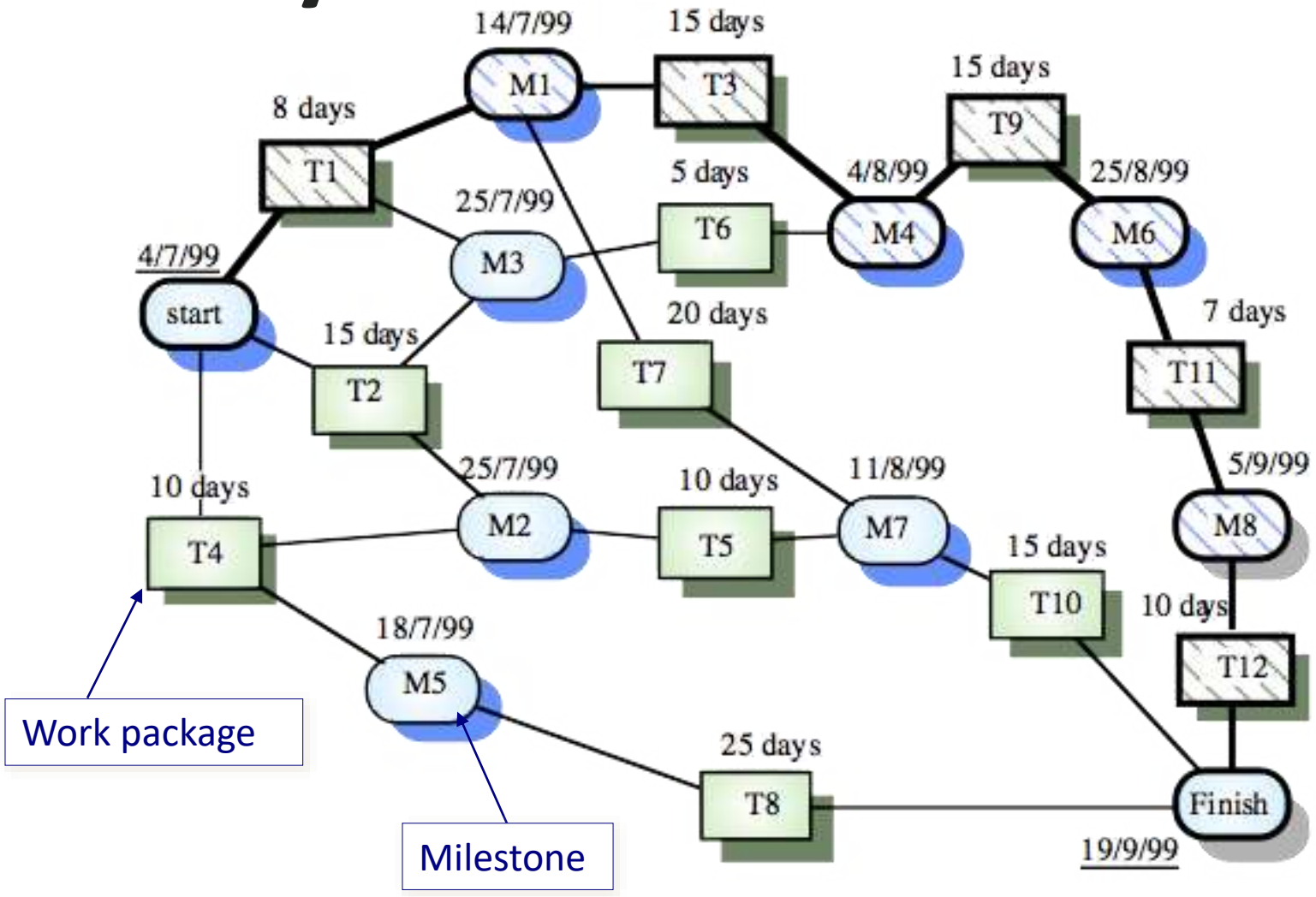


Project Manager Tasks

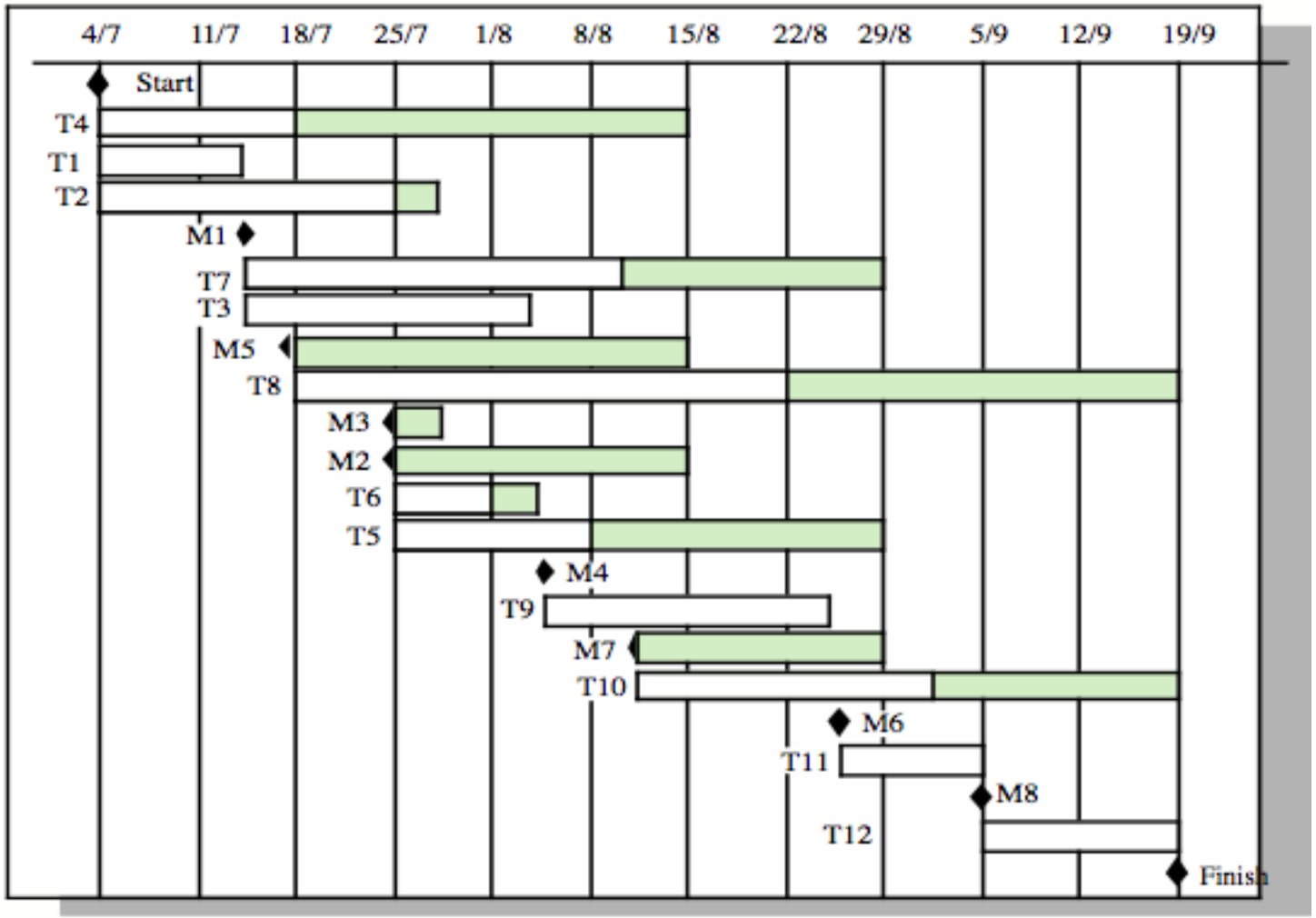
- Divide project into work packages with measurable outcomes (each 1-10 weeks)
- Estimate time and resources
- Create order and determine parallelism
- Plan buffers for anticipated and unanticipated problems
- Software available, such as, Microsoft Project, GanttProject, Kplato, etc

- Requires experience for estimation

Activity Network



Gantt Diagrams



Replanning

- Inaccurate time predictions are normal
  -> Update schedule

Task	Planned	Actual	Difficulty/Risk	Responsibility	Completed
Evaluierung	11d	11d	1	A,M	100%
Spezifikation	11d	11d	1	A,M	100%
Entwicklung	50d	77d	2		94%
- Back-End	11d	13d	1	A,M	100%
- Corese	15d	41d	3	A	80%
- MySQL	10d	32d	2	M	100%
- XML	10d	27d	2	M	90%
- Front-End	8d	18d	2	A	100%
Testing	8d	5d	1	A,M	80%
Dokumentation	20d	12d	1	A,M	100%

Reasons for Missed Deadlines

- Insufficient staff (illnesses, staff turnover, ...)
- Insufficient qualitication
- Unanticipated difficulties
- Unrealistic time estimations
- Unanticipated dependencies
- Changing requirements, additional requirements
- Especially in student projects
  - Underestimated time for learning technologies
  - Uneven work distribution

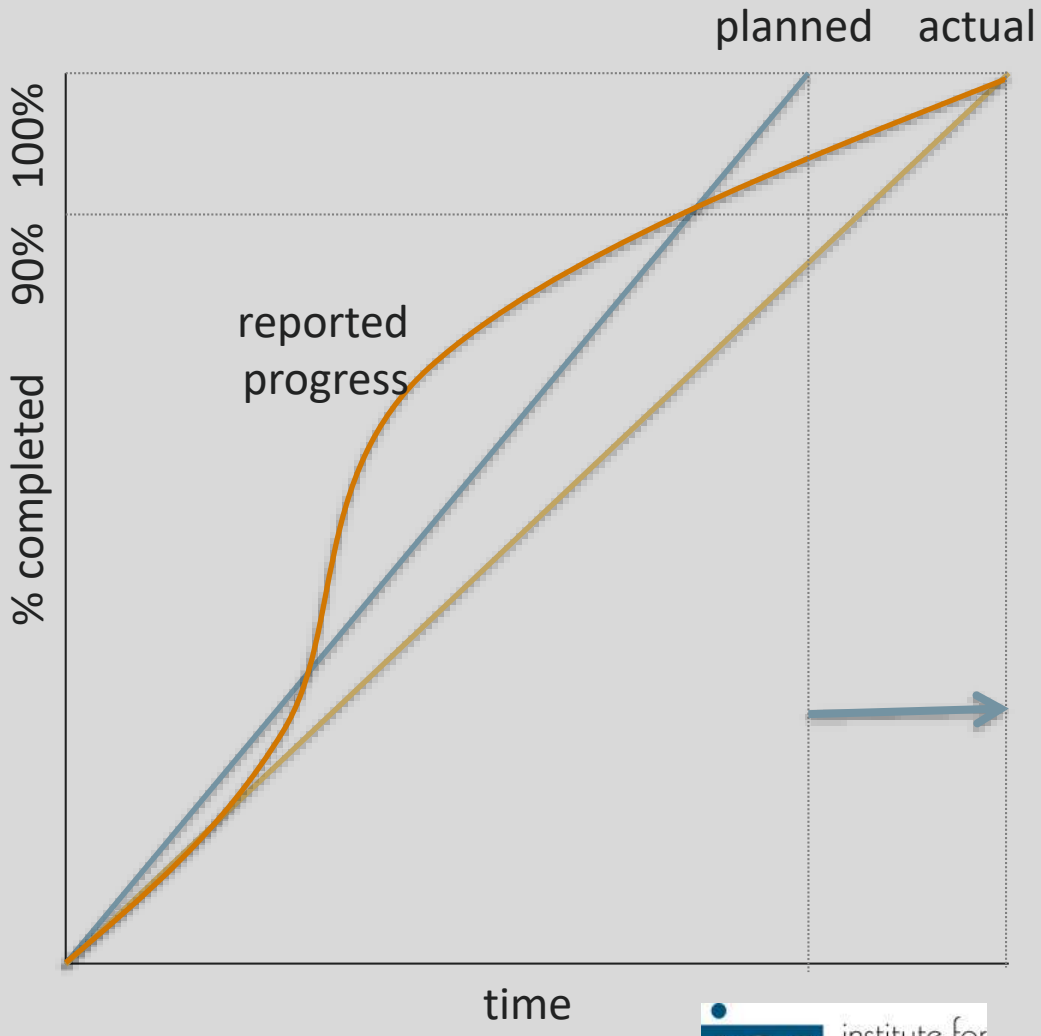
Recognize Scheduling Issues Early

- Monitoring and formal reporting necessary
  - Establish who, when, what
  - Compare planned/actual data
- Measurable milestones
- Outdated schedules no meaningful management mechanism

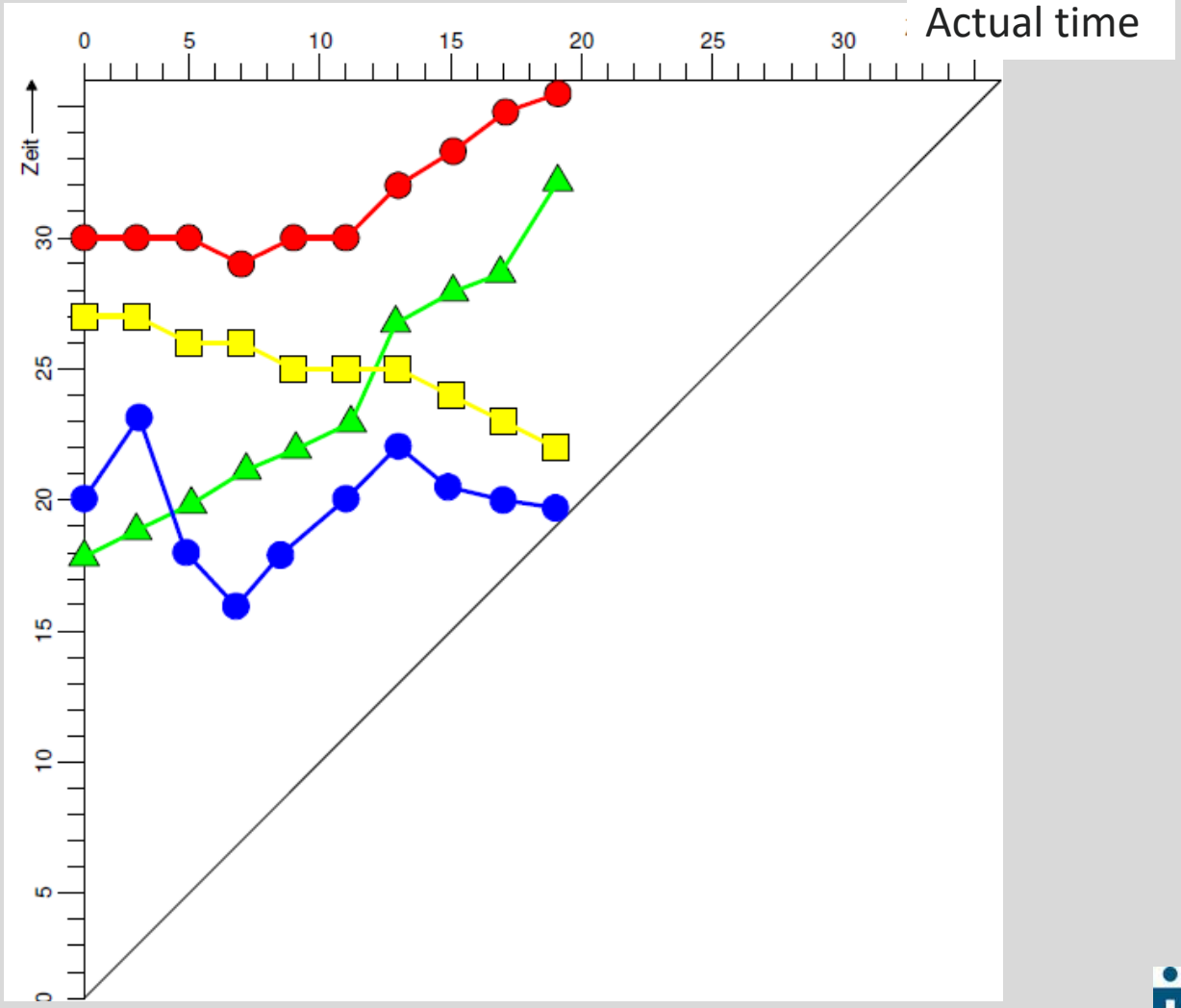
Almost Done Problem

- Last 10% of work -> 40% of time (or 20/80)

- Make progress measureable
- Avoid depending entirely on developer estimations



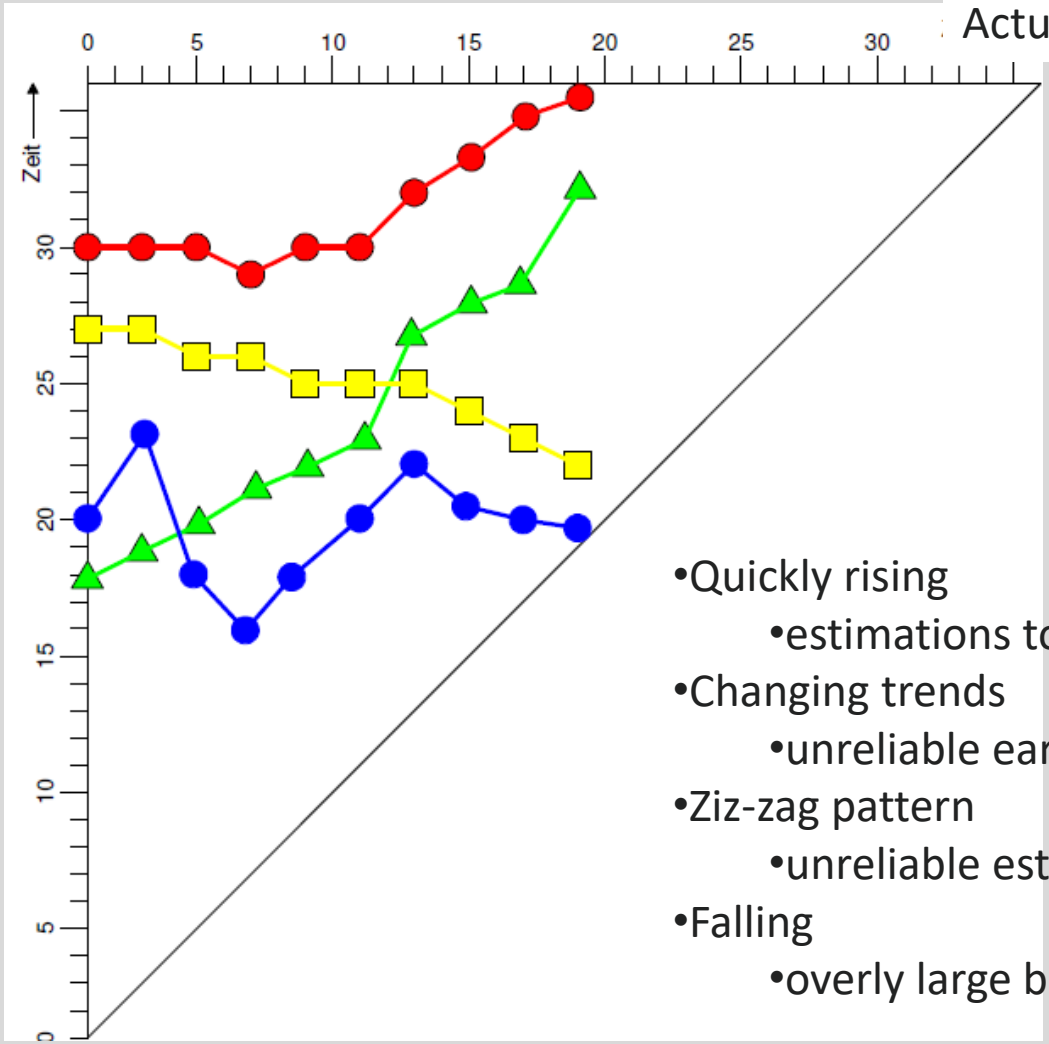
Milestone Trend Analysis



Milestone Trend Analysis

Actual time

Estimated completion time



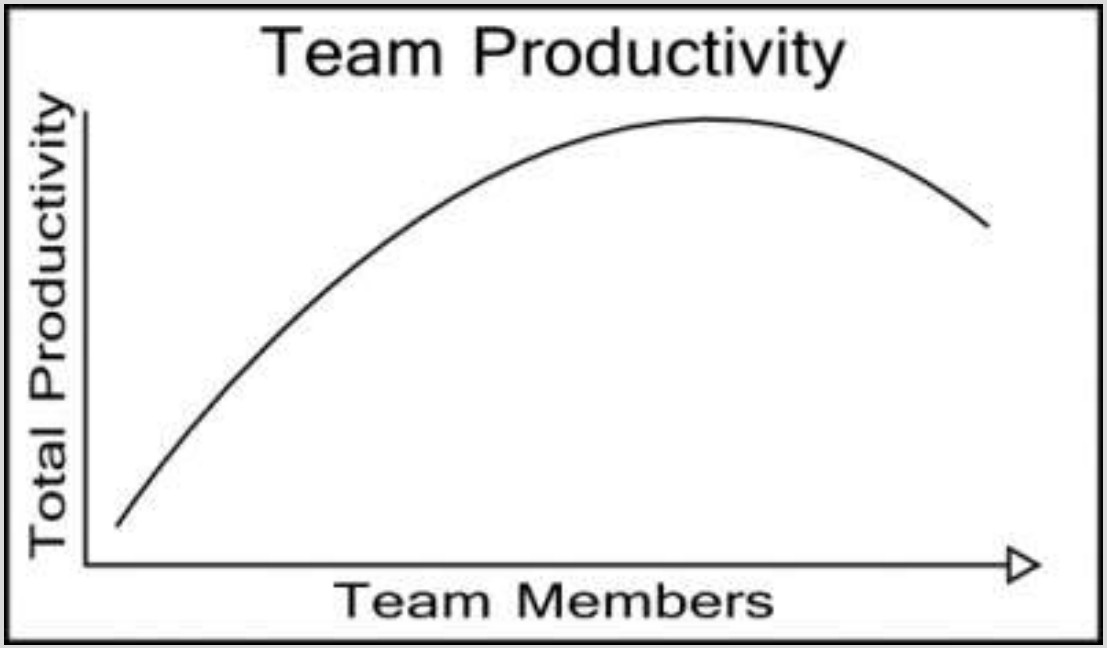
- Quickly rising
  - estimations too optimistic
- Changing trends
  - unreliable early estimations
- Ziz-zag pattern
  - unreliable estimations
- Falling
  - overly large buffers

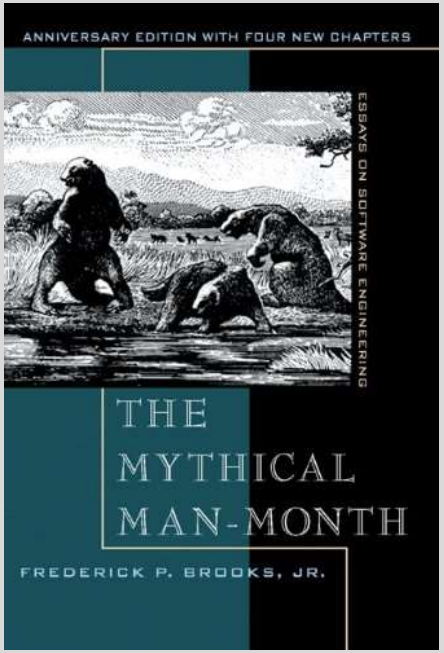
Mitigation strategies

- Additional personal
  - especially experts for specific tasks
- Temporarily increasing hours of work (overtime, vacation freeze)
  - short-term solution only
- Improve tooling, methods, and processes
- Buy, contract, off shore
- Renegotiate / reduce functionality
  - Set priorities, incremental deployment
  - Move deadline
- Avoid: less testing/quality assurance

Team productivity

- Brook's law: Adding people to a late software project makes it later.





Teamwork (Student Teams)

(more on teams in real projects later in the course)

Expectations

- Meet initially and then regularly
- Review team policy
- Divide work and integrate
- Establish a process
- **Set and document clear responsibilities and expectations**
  - Possible Roles: Coordinator, Scribe, Checker, Monitor
  - Rotate roles every assignment
- Every team member should understand the entire solution

Team Policies

- see document

- Make agreements explicit and transparent
- Most teams will encounter some problem

Dealing with problems

- Openly report even minor team issues in individual part of assignments
- In-class discussions and case studies
- Additional material throughout semester
- We will attend one team meeting

Planning and In-Team Communication

- Asana, Trello, Microsoft Project, …
- Github Wiki, Google docs, …
- Email, Slack, Facebook groups, …

Homework and Readings

Reading assignment

- Prepare a case discussion about Healthcare.gov
  - Read the TIME Magazine article “Code Red”
  - Explore Wikipedia, Youtube, other resources
  - Look at the webpage itself
- Think about what went wrong and why (both technical and nontechnical issues), how it could be fixed or prevented in future projects, …

Homework 1

- Team assignment
- Build grad school application system
- Process focus
  - Estimate and track time
  - Plan process and observe deviations
  - Analyze risks

Application Summary

The table below lists what is necessary to apply for admission to graduate study in the Department of Electrical and Computer Engineering at Carnegie Mellon.

If you have questions about the application process, please check the frequently asked questions page before submitting an email. If the FAQ does not answer your question, please contact the ECE Admissions Office. If you have technical questions or concerns regarding the online application web site, contact the ECE Web Team.

Program Selection	
Change Program Selection	1. **M.S. in ECE** • Plttsburgh, PA More Information 2. **M.S. SV** • Silicon Valley, CA More Information
Required Items	
Application for Admission and Financial Aid	Complete
Statement of Purpose	Open
Technical Areas of Interest Matrix	Open

Application Checklist

The table contains both the items that can be completed and submitted on-line and a checklist of supplementary materials that need to be mailed in.

For your application to be eligible for consideration you must complete all of the required items in the list. Remember, you can save your materials to work on at a later time.

If the status reads "Complete" or "Closed" for all of the required items in the checklist, you have finished the application process.

Further Reading

- McConnell. Software Project Survival Guide. Microsoft Press 1998, Chapter 3
- Sommerville. Software Engineering. 8th Edition. Addison-Wesley 2007. Chapters 5 "Project Planning" and 26 "Software Cost Estimation"