Software Development Process

Charlie Garrod **Michael Hilton**

School of Computer Science
institute for SOFTWARE RESEARCH

Administrivia

- Homework 6 Due Today – Thursday Dec 7th
- Final Exam Review: Dec 13th, 2-4pm Wean 5409
- Final Exam: Dec 15th, 5:30-8:30pm Wean 7500
- Faculty Course Evaluations
- https://www.ugrad.cs.cmu.edu/ta/F17/feedback/
- https://cmu.smartevals.com/

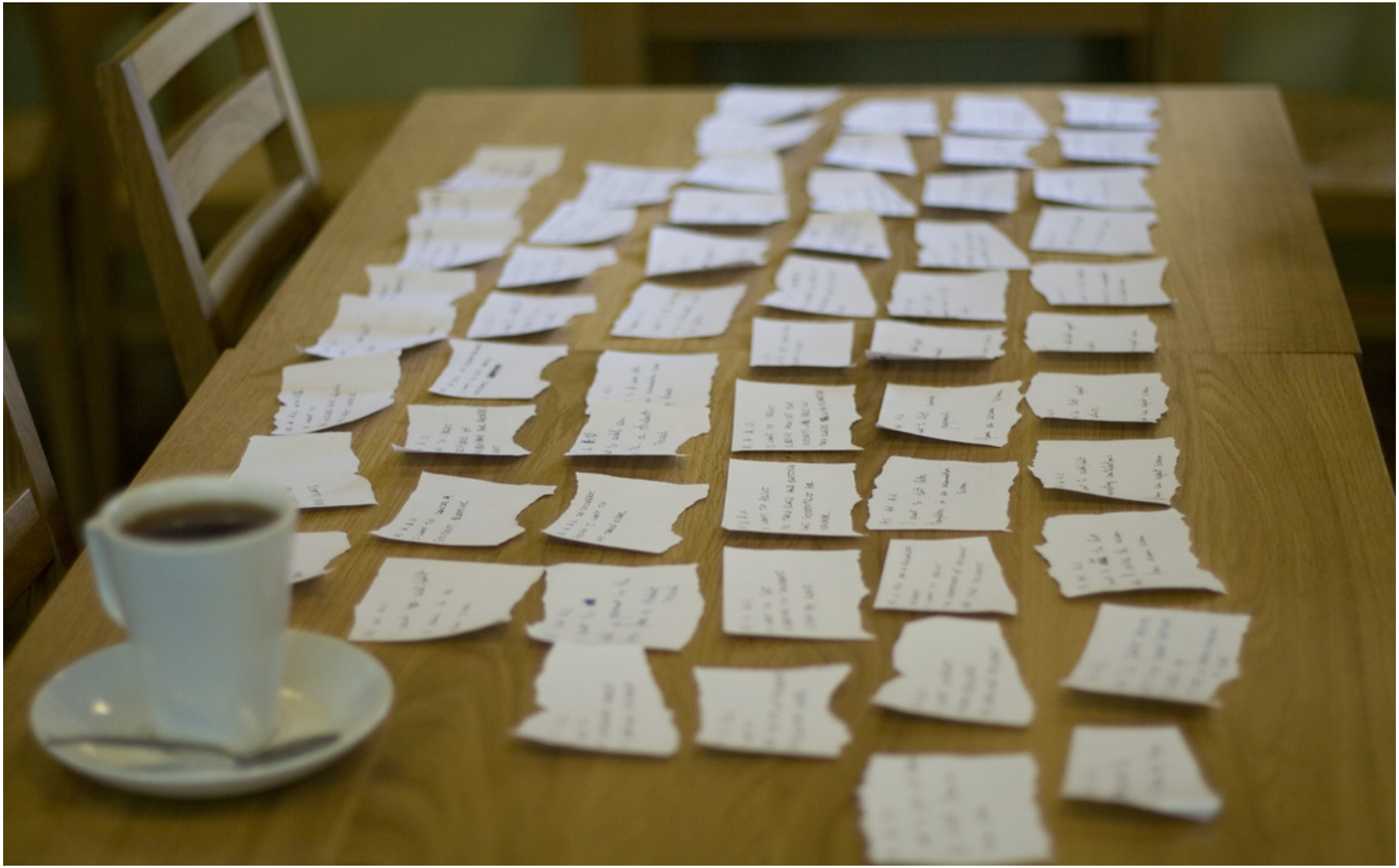
Last Time:

- Test Driven Development

How do we know what to build?

USER STORIES

User Stories Cards



https://www.flickr.com/photos/jakuza/2728096478

User Story Format

- Three C's
  - The Card
  - The Conversation
  - The Confirmation

The Card

- A natural language description of one or more features of a system.
- Should fit on a single index card
- Format:
  - As a <ROLE>,
  - I want <FUNCTION>
  - so that <VALUE>.

The Conversation

- A dialog between everyone working on the project and the client (stakeholder)
- Split up Epic Stories if needed
- Adds more details to user stories
- Can be a list, or bullet points

The Confirmation

- A description of how we will know when this user story is done
- A test that will show when the task is completed
- Could be automated, or a script

Exercise

- https://vimeo.com/41800652
- Each person should develop one user story

How do we know if we have good user stories?

I-N-V-E-S-T

INVEST

- Independent
  - Schedule in any order
  - Not overlapping in concept
  - Not always possible
- Negotiable
  - Details to be negotiated during development
  - Good story captures the essence, not the details
- Valuable
  - This story needs to have value to someone
  - Especially relevant when splitting up issues
- Estimable
  - Helps keep the size small
  - Ensure we negotiated correctly

INVEST cont…

- Small
  - Fit on a 3x5 card
  - At most two person-weeks of work
  - Too big == unable to estimate
- Testable
  - Ensures understanding of task
  - We know when we can mark task “Done”
  - Unable to test == do not understand

PROCESS

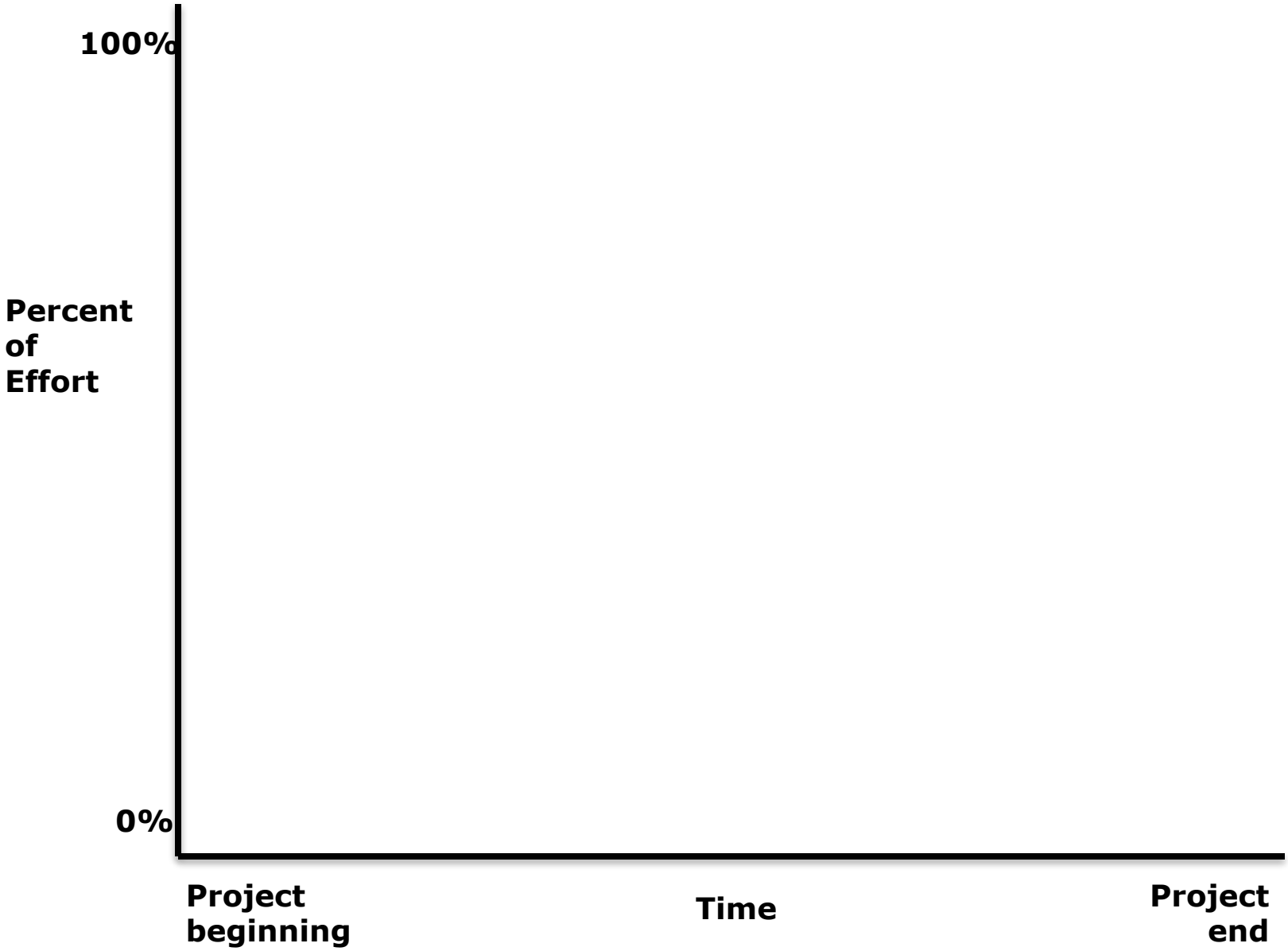
How to develop software?

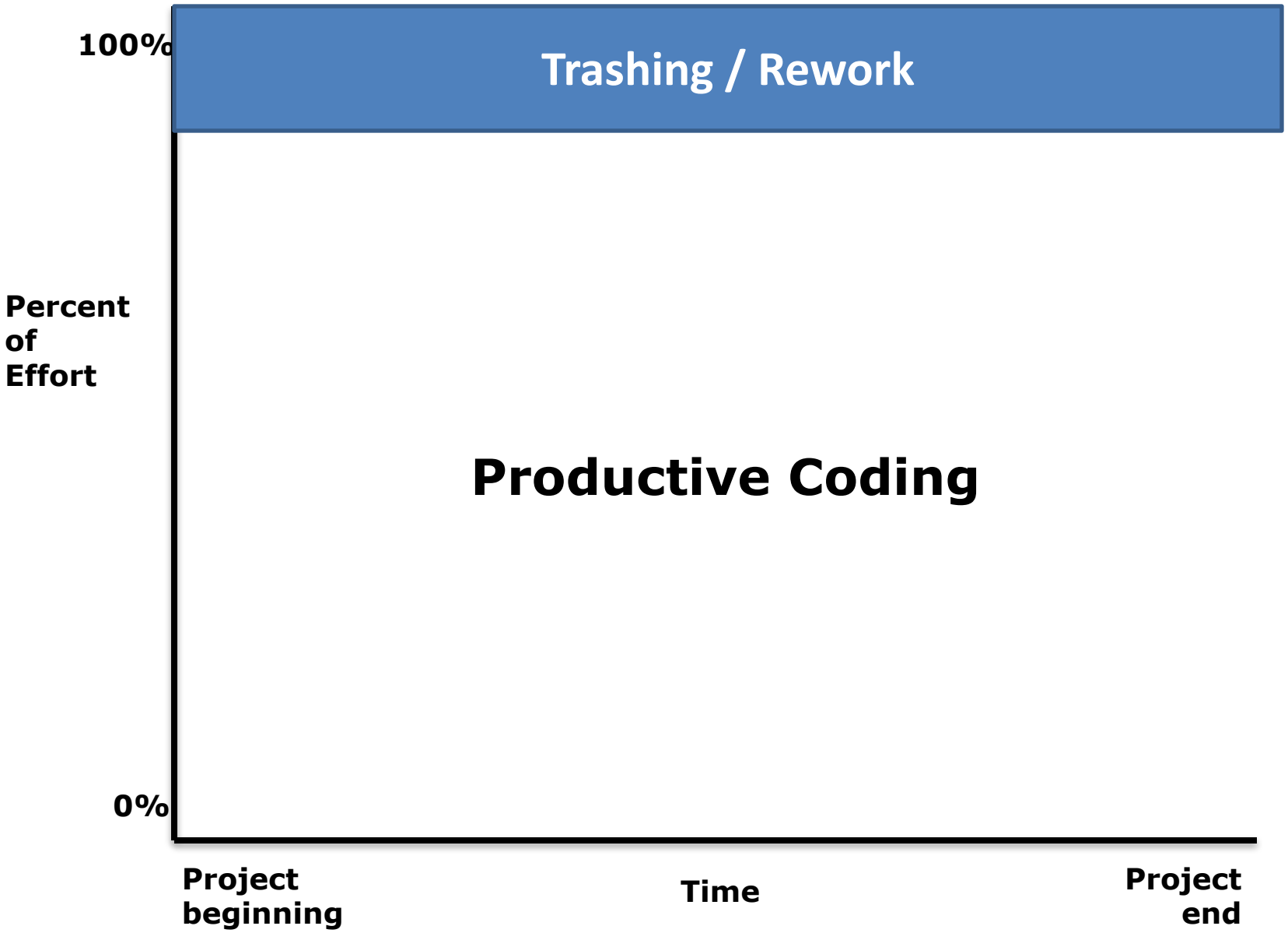
1. Discuss the software that needs to be written
2. Write some code
3. Test the code to identify the defects
4. Debug to find causes of defects
5. Fix the defects
6. If not done, return to step 1

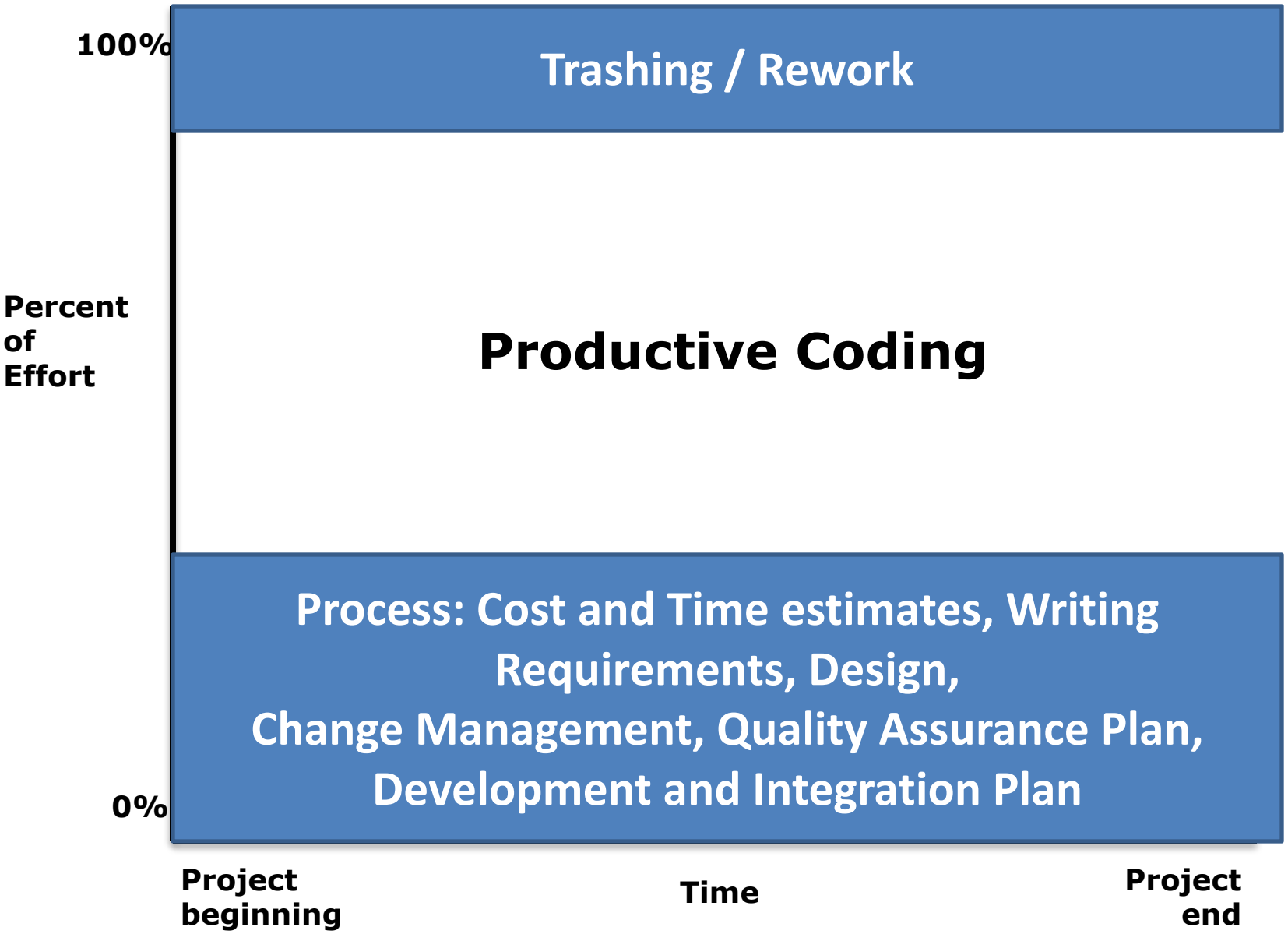
Software Process

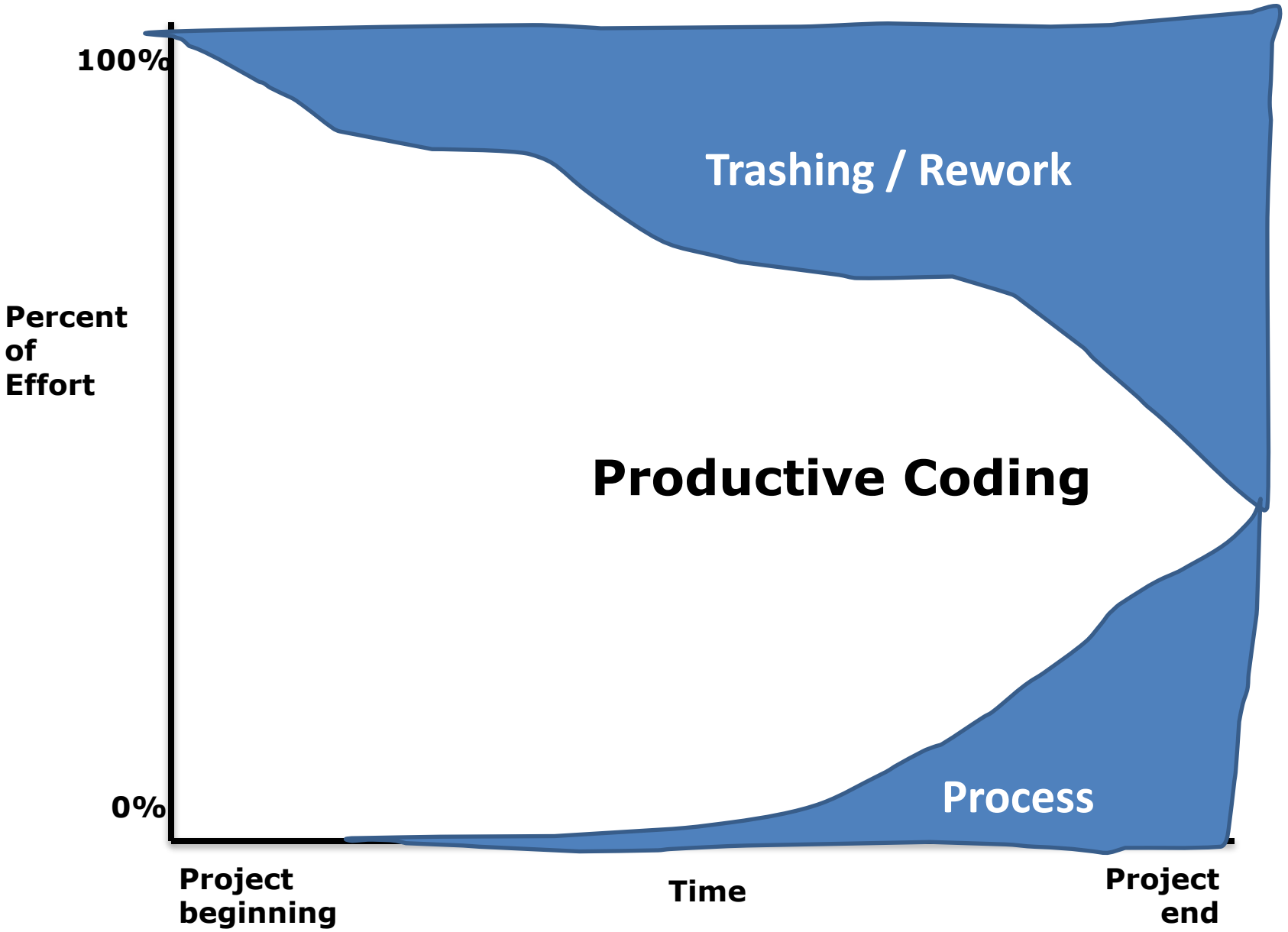
"The set of activities and associated results that produce a software product"

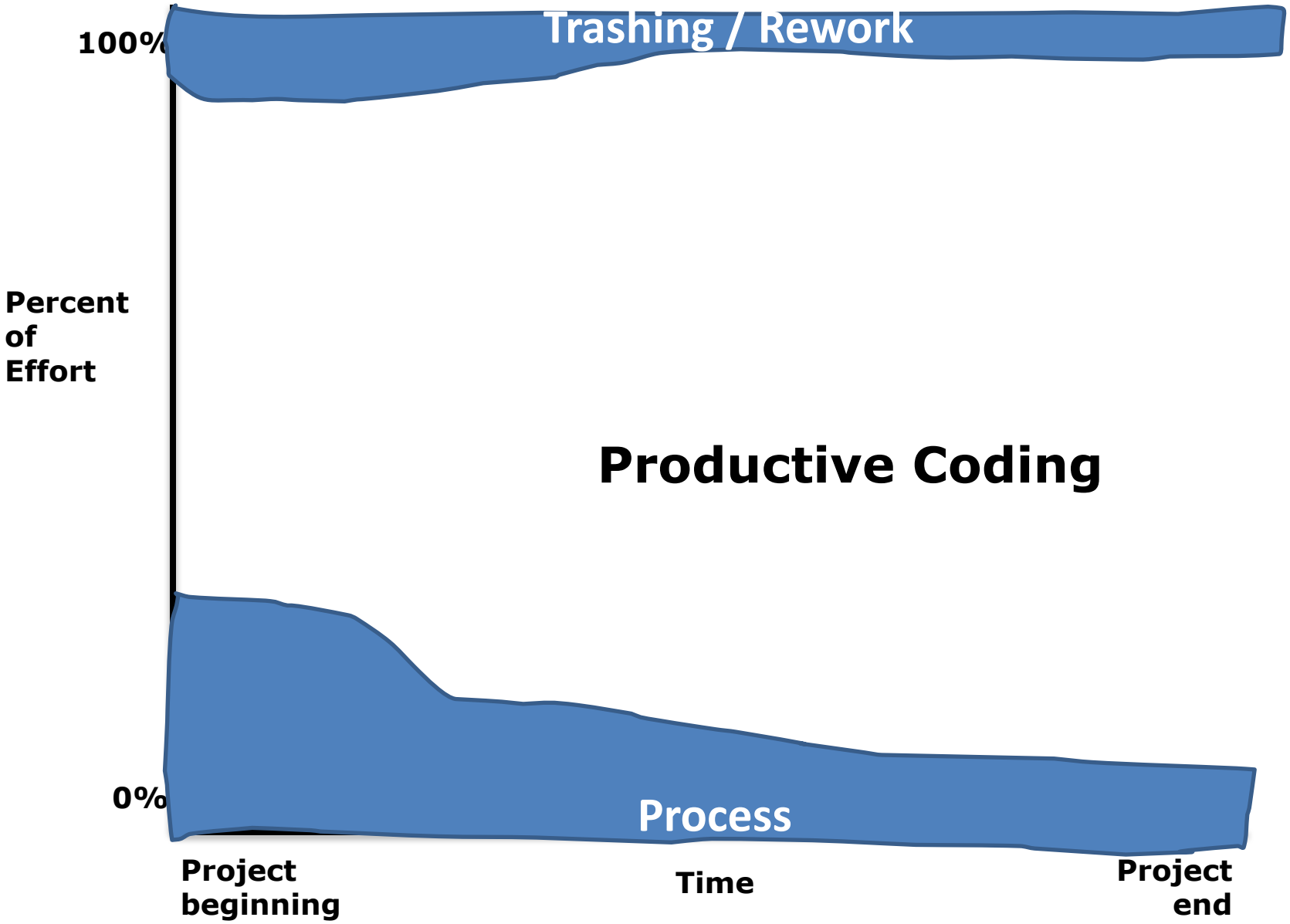
What makes a good process?




100%
Trashing / Rework
Percent of Effort
Productive Coding
0%
Project beginning
Time
Project end






100%
Trashing / Rework
Percent of Effort
Productive Coding
0%
Process
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.

RETROSPECTIVES

Retrospective

- A look back with the intention to Inspect and Adapt
- Without introspection there can be no improvement
- We continually improve our processes by inspecting and adapting.

Retrospective Questions

- What went well?
- What went poorly?
- What could we do differently to improve?

Summary: take 17-313!

- Software Engineering in practice requires consideration of numerous issues---technical and social---above the level of individual class design/implementation.
- Do you think this is interesting? 17-313, Foundations of Software Engineering is offered in the Fall.
- And consider the undergraduate SE minor!