[CS3704] Software Engineering

Dr. Chris Brown

Virginia Tech

9/8/2023

Discussion Presentation

SE Research Overview
Discussion Presentation Details
Example Presentation and Activity

Announcements

- **Sign up for a discussion presentation**
  - Today in class
- **HW1 due tonight by 11:59pm!**
- **PM0 due Monday by 11:59pm**
  - Teammate preferences and project questions
- **No class on Monday**
  - Career Fair 💼
  - The career fair does *not* count for Ut Prosim points for class

Project Questions so far

How can we access the list of examples for the project?
https://github.com/CS3704-VT/Course/blob/main/Project/IDEAS.md

Would a project that primarily supports other work, but is still usable by Software Engineers for Software Engineering work fulfill these requirements? Maybe, it depends

Currently, we have three teammates, is another required? Probably

What is software engineering?

A discipline that encompasses:

- the ***process*** of software development;
- ***methods*** for software analysis, design, construction, testing, and maintenance; and
- ***tools*** that support the processes and the methods.

[Pressman]

Software Engineers

A person who applies a systematic engineering approach to the design, development, testing, and maintenance of computer software.

- Also known as developer, programmer,...

Traits of Successful Software Engineers

- Sense of individual responsibility
  - Do what needs to be done in an overriding effort to achieve a successful outcome
- Awareness of stakeholder needs
  - Observe the environment in which people work and adapt his/her behavior
- Honest about design flaws and offer constructive criticism
  - Be realistic and truthful

Traits of Successful Software Engineers

- Resilient under pressure
  - Manage the pressure/chaos which comes in many forms: changing requirements, demanding stakeholders, unrealistic manager
- Attention to details
  - Consider the technical decisions against broader criteria
- Pragmatic
  - SE is a discipline to be adapted based on circumstances

Attributes of Effective SE Teams

- Sense of purpose
  - Everyone agrees on the goal
- Sense of involvement
  - Everyone feels that their skillset and contributions are valued
- Sense of trust
  - Everybody should trust the skills and competence of their peers and their managers
- Sense of improvement
  - Periodically reflect to think about ways for improvement
- Diversity of team members skills, backgrounds,...

Why do we need a team?

- Software is to big and complex to be constructed by a single person.

Possible team crises:

– Team member leaves
– Team member laziness (or incompetence)
– Team member is anti-social
– Machine problems
– Scheduling difficulties…

Make sure to adapt for your course project team!

Avoid Team Toxicity

- Frenzied work atmospheres
  - Define goals and objectives
- Frustration that causes friction
  - Make decisions as a team as much as possible
- Fragmentation and poor coordination
  - understanding the tasks to be done, the people doing the work, etc.
- Unclear definition of roles
- Continuous and repeated exposure to failure

Current SE Problems

- Software is too expensive and takes too long to build
    - Frequently over budget and over time!
- Low software quality
    - 3.6 billion users, $1.7 trillion caused by bugs [Tricentis, 2017]
- Software is more complex to support and maintain
- More users = more difficult to scale applications
- Failing to meet requirements
- Lack of diversity in software development teams
- Inadequate testing and security
- Ethical decisions in software engineering
- ***What else???*** **[HW1]**

Software Engineering?

“ ‘It’s Engineering... but not as we know it’… Software Engineering - solution to the software crisis, or part of the problem? ”

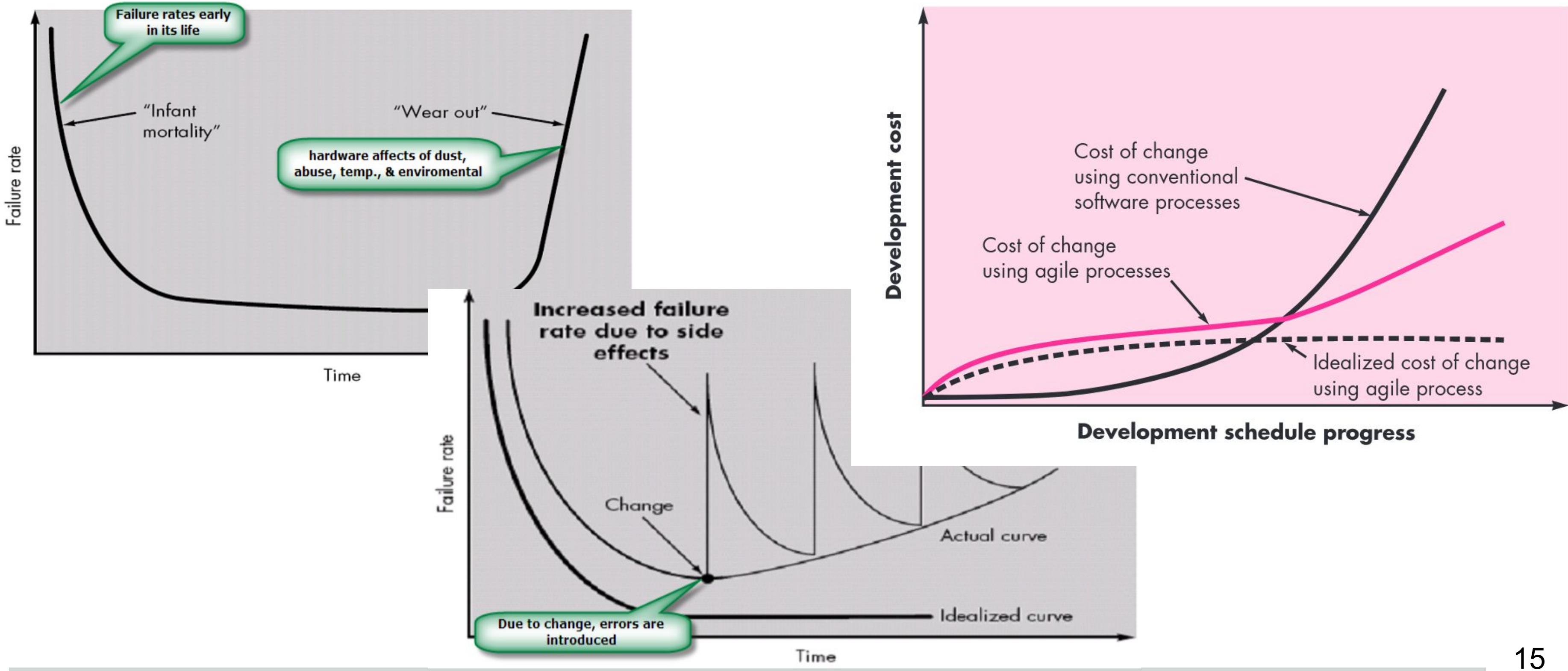
[Bryant]

SE Research

Software engineering research seeks to create, understand, and evaluate the strengths and weaknesses of SE tools and practices, with the goal of improving the lives and evaluating the work of software engineers.

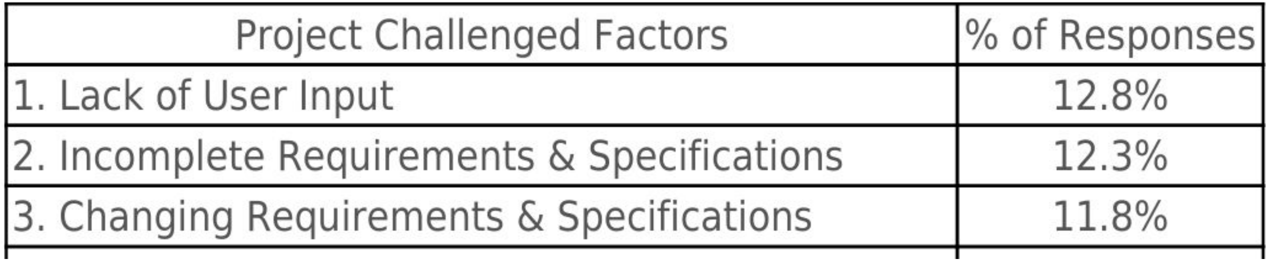
Examples we’ve seen in class so far include…

SE Research: Costs



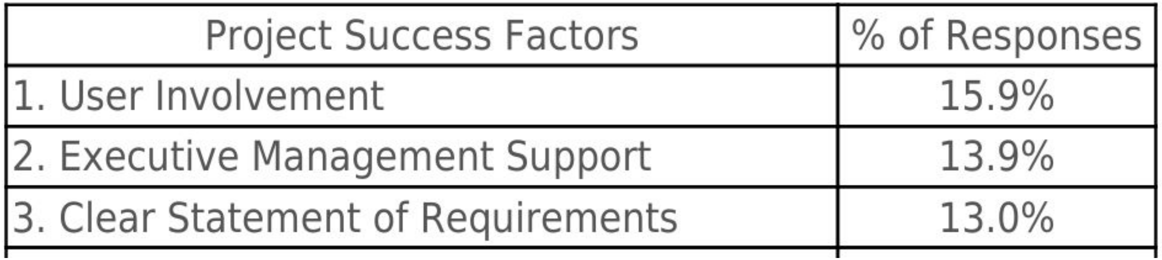
SE Research: Project Failure

Top 3 reasons for project failure:



Project Challenged Factors	% of Responses
1. Lack of User Input	12.8%
2. Incomplete Requirements & Specifications	12.3%
3. Changing Requirements & Specifications	11.8%

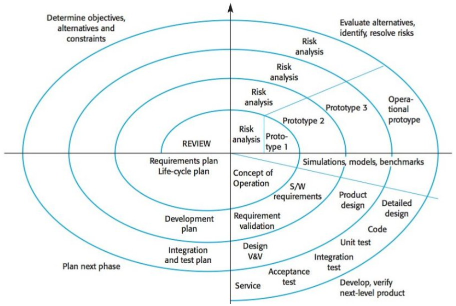
Top 3 reasons for project success:



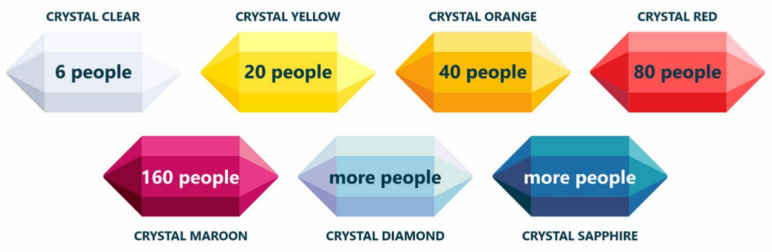
Project Success Factors	% of Responses
1. User Involvement	15.9%
2. Executive Management Support	13.9%
3. Clear Statement of Requirements	13.0%

[Standish group, 1995]

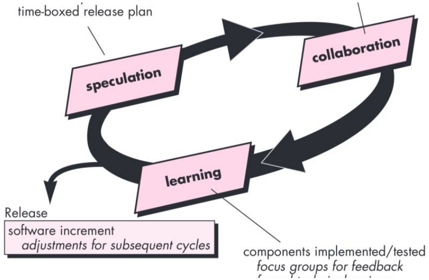
SE Research: Process/Frameworks



"A Spiral Model of Software Development and Enhancement" [Boehm, 1988]



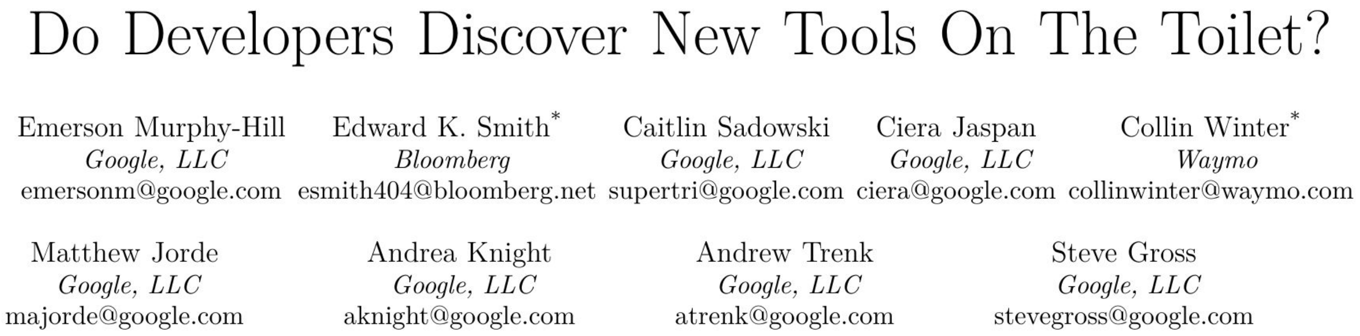
"Crystal clear: A Human-powered methodology for small teams" [Cockburn, 2004]


time-boxed release plan
speculation
collaboration
learning
Release
software increment
adjustments for subsequent cycles
components implemented/tested
focus groups for feedback

"Adaptive Software Development: An Evolutionary Approach to Managing Complex Systems [Highsmith, 2000]

SE Research (cont.)

But, SE research can also be irrelevant and useless for actual software engineers and their work…



Do Developers Discover New Tools On The Toilet?

Emerson Murphy-Hill
Google, LLC
emersonm@google.com

Edward K. Smith*
Bloomberg
esmith404@bloomberg.net

Caitlin Sadowski
Google, LLC
supertri@google.com

Ciera Jaspan
Google, LLC
ciera@google.com

Collin Winter*
Waymo
collinwinter@waymo.com

Matthew Jorde
Google, LLC
majorde@google.com

Andrea Knight
Google, LLC
aknight@google.com

Andrew Trenk
Google, LLC
atrenk@google.com

Steve Gross
Google, LLC
stevegross@google.com

- Yes, they do 💩

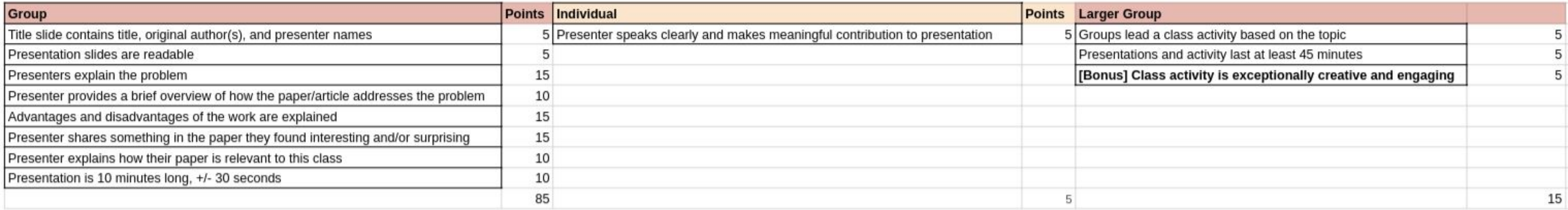
Research Discussion

- Each student will present one SE-related research paper or article as a group of five.
  - You may send a replacement to me to be approved *at least one week before your presentation date*.
- All groups (n = 3) presenting on a specific day will lead a discussion/activity for class.

Learning Outcome:

- **Discuss research questions and studies related to software engineering**

Rubric



Group	Points	Individual	Points	Larger Group	
Title slide contains title, original author(s), and presenter names	5	Presenter speaks clearly and makes meaningful contribution to presentation	5	Groups lead a class activity based on the topic	5
Presentation slides are readable	5			Presentations and activity last at least 45 minutes	5
Presenters explain the problem	15			**[Bonus] Class activity is exceptionally creative and engaging**	5
Presenter provides a brief overview of how the paper/article addresses the problem	10				
Advantages and disadvantages of the work are explained	15				
Presenter shares something in the paper they found interesting and/or surprising	15				
Presenter explains how their paper is relevant to this class	10				
Presentation is 10 minutes long, +/- 30 seconds	10				
	85		5		15

Tips

- Read the paper thoroughly, but don’t try to understand all of the details
- Practice your talk ahead of time (Stay on time!)
- Always start with the *problem*
- Be creative in your discussion or activity
- Grade is based on individual presentation, group discussion, and overall class activity
- Your research discussion should go something like this…

Late Warm-Up

- Sign up for a research discussion talk.
- Discuss with a partner/small group why you selected the topic you chose.

What Makes A Great Software Engineer?

Authors: Paul Luo Li, Amy J. Ko, and Jiamin Zhu

Presenter: Chris Brown

International Conference on Software Engineering (ICSE) 2015

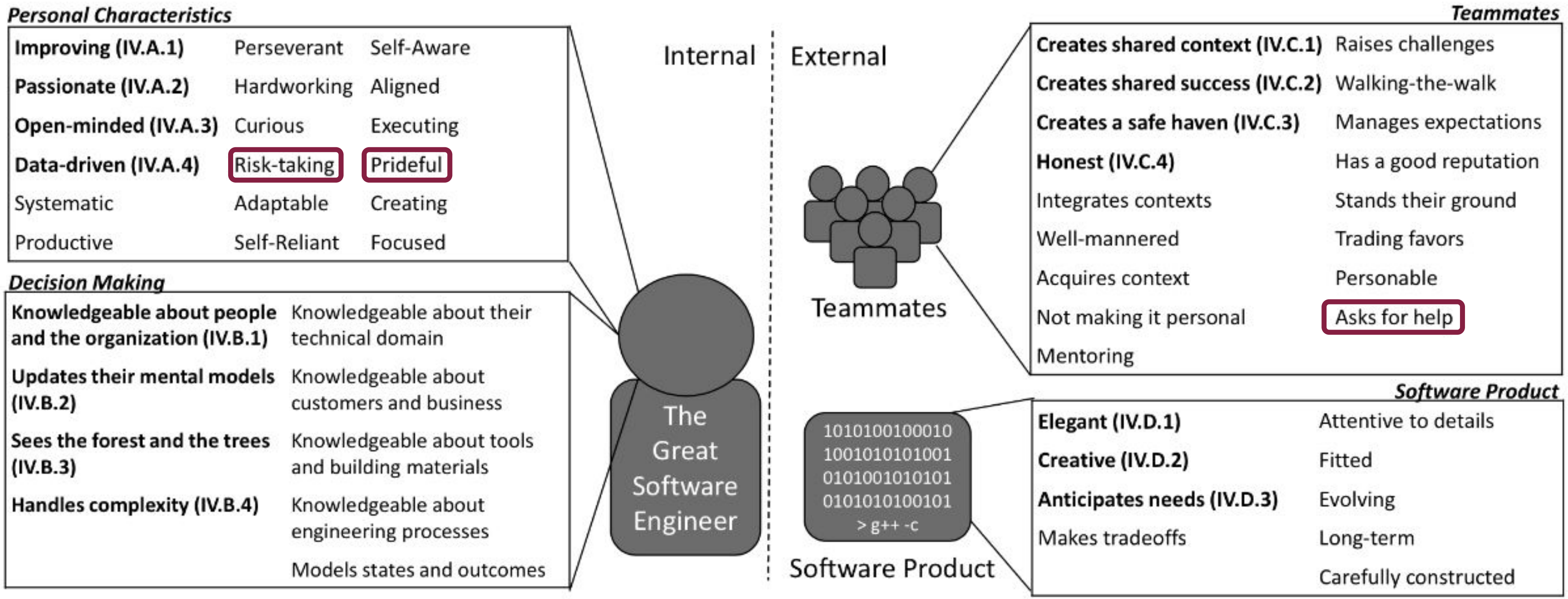
Problem

- Good software engineers are essential for developing high-quality software.
  - Companies want to hire them
  - Universities want to train them
  - Students/Novices want to be them
- **But, software engineers are difficult to evaluate!**
  - Technical skills, and beyond

Evaluation

- Interviewed 59 experienced software engineers at Microsoft.
  - Questions were mostly reflective
    - (i.e. *“Think back to someone you’ve worked with that you that was a great software engineer. What were some attributes that made the person ‘great’ in your mind?”*)
- Analyzed responses to derive 53 attributes of great software engineers.

Results



Personal Characteristics

Improving (IV.A.1)	Perseverant	Self-Aware
Passionate (IV.A.2)	Hardworking	Aligned
Open-minded (IV.A.3)	Curious	Executing
Data-driven (IV.A.4)	Risk-taking	Prideful
Systematic	Adaptable	Creating
Productive	Self-Reliant	Focused

Decision Making

Knowledgeable about people and the organization (IV.B.1)	Knowledgeable about their technical domain
Updates their mental models (IV.B.2)	Knowledgeable about customers and business
Sees the forest and the trees (IV.B.3)	Knowledgeable about tools and building materials
Handles complexity (IV.B.4)	Knowledgeable about engineering processes
	Models states and outcomes

Teammates

Creates shared context (IV.C.1)	Raises challenges
Creates shared success (IV.C.2)	Walking-the-walk
Creates a safe haven (IV.C.3)	Manages expectations
Honest (IV.C.4)	Has a good reputation
Integrates contexts	Stands their ground
Well-mannered	Trading favors
Acquires context	Personable
Not making it personal	Asks for help
Mentoring	

Software Product

Elegant (IV.D.1)	Attentive to details
Creative (IV.D.2)	Fitted
Anticipates needs (IV.D.3)	Evolving
Makes tradeoffs	Long-term
	Carefully constructed

Fig. 1. Model of attributes of great software engineers, with attributes we discuss in detailed in bold.

Surprises

Risk-taking—willing to go into high-value areas even though they may not have knowledge or expertise (e.g. new technologies).

Prideful—taking pride in oneself and ones' product; letting their output be a reflection of their skills and trying their best to deliver.

Asks for help—finding and engaging others with needed knowledge and information.

Advantages of Research

- First study to analyze what makes a great software engineer.
- Authors provide implications for research, new software engineers, management, and education.

Advantages of Research

Researchers: Explore methods to measure *internal* attributes and develop tools and processes to increase *external* attributes.

Novice Software Engineers: Set of attributes to emulate, aspire to achieve, and to seek in potential mentors.

Managers: *“Walk the walk”*, create an environment and culture to foster these attributes with your development team.

Educators: Examine teaching methods and curricula, create a classroom environment and culture to foster these attributes.

Limitations

- Only conducted at Microsoft
    - *What about developers at other companies?*
- Does’t include quantitative data on results
    - i.e. *how many participants mentioned each attribute?*
- Lacks details on the background of participants
    - i.e. *average years of experience, demographic information, etc.*

Relevance to Class

Software engineering is a human activity, and software engineers use processes, methods, and tools to develop and maintain applications.

Today Was a Good Day: The Daily Life of Software Developers

Authors: Andre N. Meyer, Earl T. Barr, Christian Bird, and Thomas Zimmerman

Presenter: Chris Brown

IEEE Transactions on Software Engineering (TSE) 2021

Problem

- Software engineering is a complicated and chaotic process with many distractions.
- Good work days increase developer productivity, code quality, and job satisfaction.

Evaluation

Research Questions:

1. What factors influence good and typical developer workdays and how do they interrelate?
2. How do developers spend their time on a good and typical workday?*
3. What are the different types of workdays and which ones are more often good and typical?*
4. How does collaboration impact good and typical workdays?

Evaluation (cont.)

- Interviewed software engineers to discover work activities.
- Sent 37,792 surveys to developers and received 5,971 responses to characterize activities with workdays.
    - Types of workdays: Typical, Atypical, Good, and Bad

Results

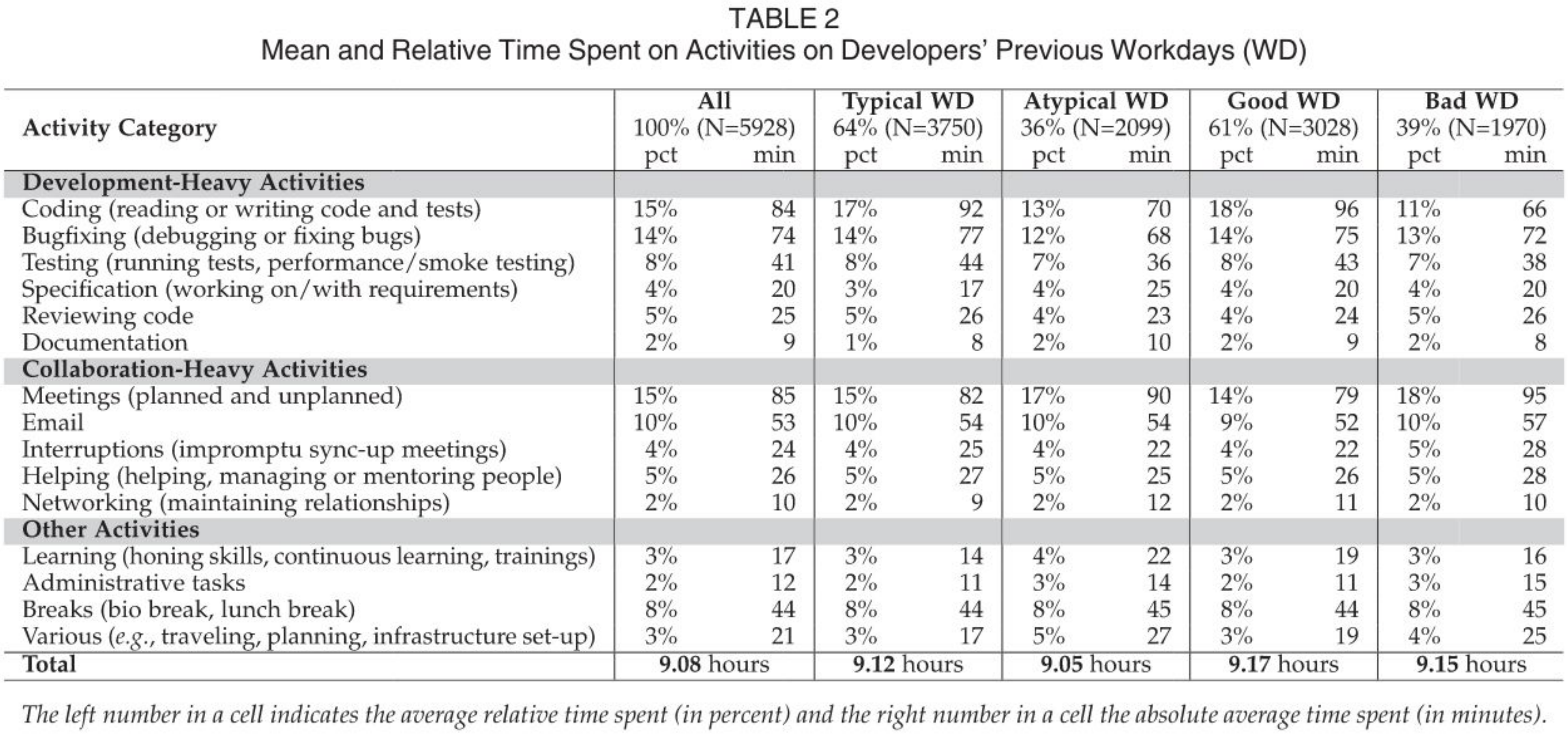


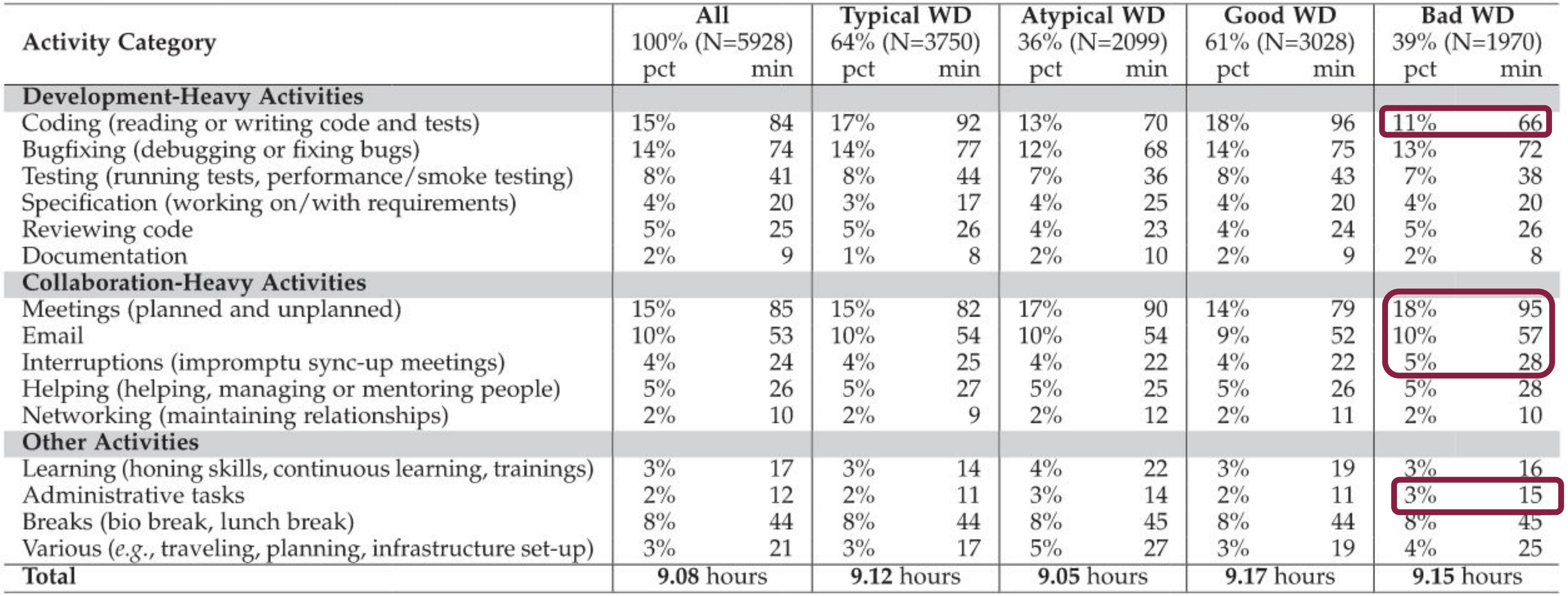
TABLE 2
Mean and Relative Time Spent on Activities on Developers' Previous Workdays (WD)

Activity Category	All		Typical WD		Atypical WD		Good WD		Bad WD	
	100% (N=5928)		64% (N=3750)		36% (N=2099)		61% (N=3028)		39% (N=1970)	
	pct	min	pct	min	pct	min	pct	min	pct	min
Development-Heavy Activities										
Coding (reading or writing code and tests)	15%	84	17%	92	13%	70	18%	96	11%	66
Bugfixing (debugging or fixing bugs)	14%	74	14%	77	12%	68	14%	75	13%	72
Testing (running tests, performance/smoke testing)	8%	41	8%	44	7%	36	8%	43	7%	38
Specification (working on/with requirements)	4%	20	3%	17	4%	25	4%	20	4%	20
Reviewing code	5%	25	5%	26	4%	23	4%	24	5%	26
Documentation	2%	9	1%	8	2%	10	2%	9	2%	8
Collaboration-Heavy Activities										
Meetings (planned and unplanned)	15%	85	15%	82	17%	90	14%	79	18%	95
Email	10%	53	10%	54	10%	54	9%	52	10%	57
Interruptions (impromptu sync-up meetings)	4%	24	4%	25	4%	22	4%	22	5%	28
Helping (helping, managing or mentoring people)	5%	26	5%	27	5%	25	5%	26	5%	28
Networking (maintaining relationships)	2%	10	2%	9	2%	12	2%	11	2%	10
Other Activities										
Learning (honing skills, continuous learning, trainings)	3%	17	3%	14	4%	22	3%	19	3%	16
Administrative tasks	2%	12	2%	11	3%	14	2%	11	3%	15
Breaks (bio break, lunch break)	8%	44	8%	44	8%	45	8%	44	8%	45
Various (*e.g.*, traveling, planning, infrastructure set-up)	3%	21	3%	17	5%	27	3%	19	4%	25
Total	9.08 hours		9.12 hours		9.05 hours		9.17 hours		9.15 hours	

The left number in a cell indicates the average relative time spent (in percent) and the right number in a cell the absolute average time spent (in minutes).

Unsurprising

TABLE 2
Mean and Relative Time Spent on Activities on Developers' Previous Workdays (WD)



	All		Typical WD		Atypical WD		Good WD		Bad WD	
Activity Category	100% (N=5928)		64% (N=3750)		36% (N=2099)		61% (N=3028)		39% (N=1970)	
	pct	min	pct	min	pct	min	pct	min	pct	min
Development-Heavy Activities										
Coding (reading or writing code and tests)	15%	84	17%	92	13%	70	18%	96	11%	66
Bugfixing (debugging or fixing bugs)	14%	74	14%	77	12%	68	14%	75	13%	72
Testing (running tests, performance/smoke testing)	8%	41	8%	44	7%	36	8%	43	7%	38
Specification (working on/with requirements)	4%	20	3%	17	4%	25	4%	20	4%	20
Reviewing code	5%	25	5%	26	4%	23	4%	24	5%	26
Documentation	2%	9	1%	8	2%	10	2%	9	2%	8
Collaboration-Heavy Activities										
Meetings (planned and unplanned)	15%	85	15%	82	17%	90	14%	79	18%	95
Email	10%	53	10%	54	10%	54	9%	52	10%	57
Interruptions (impromptu sync-up meetings)	4%	24	4%	25	4%	22	4%	22	5%	28
Helping (helping, managing or mentoring people)	5%	26	5%	27	5%	25	5%	26	5%	28
Networking (maintaining relationships)	2%	10	2%	9	2%	12	2%	11	2%	10
Other Activities										
Learning (honing skills, continuous learning, trainings)	3%	17	3%	14	4%	22	3%	19	3%	16
Administrative tasks	2%	12	2%	11	3%	14	2%	11	3%	15
Breaks (bio break, lunch break)	8%	44	8%	44	8%	45	8%	44	8%	45
Various (*e.g.*, traveling, planning, infrastructure set-up)	3%	21	3%	17	5%	27	3%	19	4%	25
Total	**9.08** hours		**9.12** hours		**9.05** hours		**9.17** hours		**9.15** hours	

The left number in a cell indicates the average relative time spent (in percent) and the right number in a cell the absolute average time spent (in minutes).

Surprises

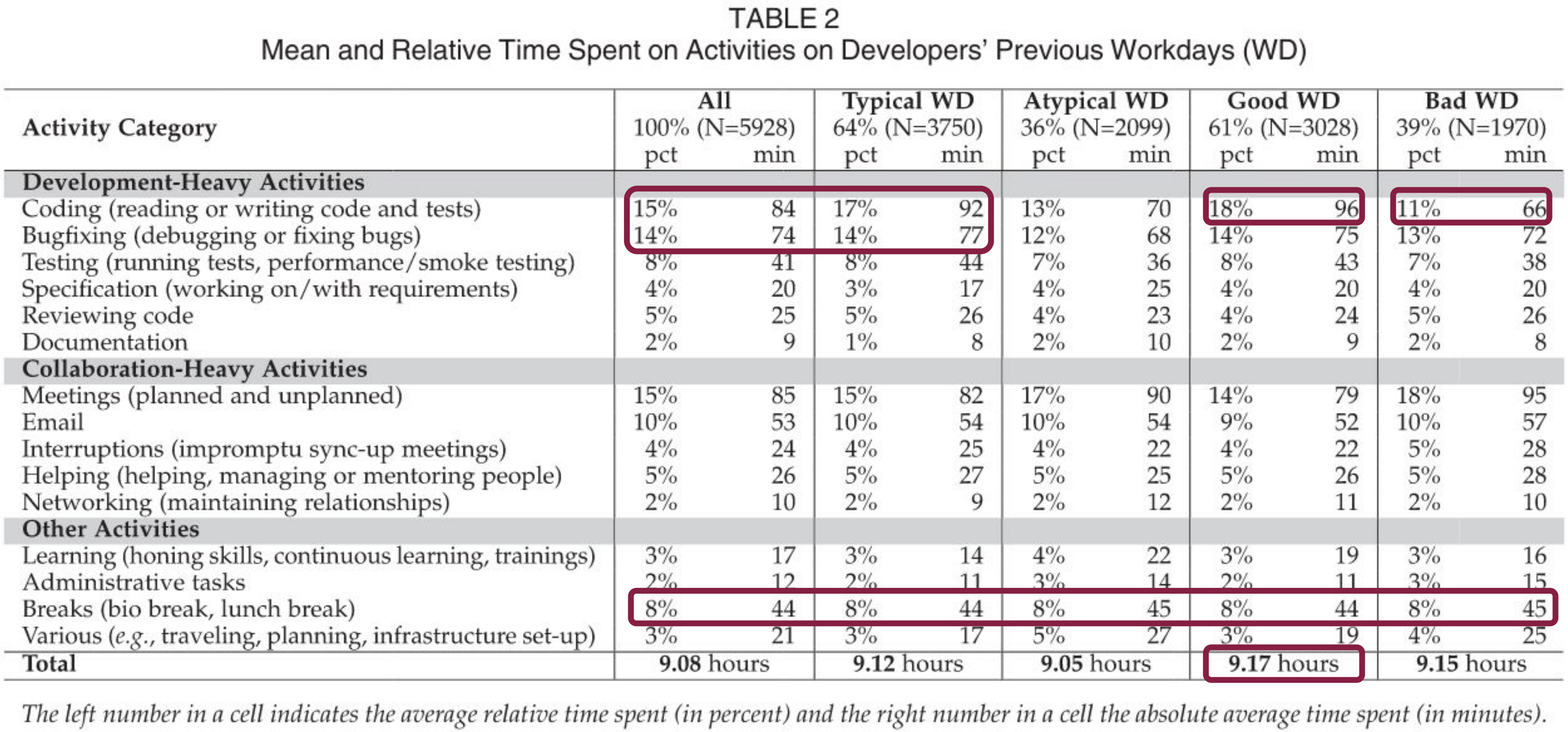


TABLE 2
Mean and Relative Time Spent on Activities on Developers' Previous Workdays (WD)

	All		Typical WD		Atypical WD		Good WD		Bad WD	
Activity Category	100% (N=5928)		64% (N=3750)		36% (N=2099)		61% (N=3028)		39% (N=1970)	
	pct	min	pct	min	pct	min	pct	min	pct	min
Development-Heavy Activities										
Coding (reading or writing code and tests)	15%	84	17%	92	13%	70	18%	96	11%	66
Bugfixing (debugging or fixing bugs)	14%	74	14%	77	12%	68	14%	75	13%	72
Testing (running tests, performance/smoke testing)	8%	41	8%	44	7%	36	8%	43	7%	38
Specification (working on/with requirements)	4%	20	3%	17	4%	25	4%	20	4%	20
Reviewing code	5%	25	5%	26	4%	23	4%	24	5%	26
Documentation	2%	9	1%	8	2%	10	2%	9	2%	8
Collaboration-Heavy Activities										
Meetings (planned and unplanned)	15%	85	15%	82	17%	90	14%	79	18%	95
Email	10%	53	10%	54	10%	54	9%	52	10%	57
Interruptions (impromptu sync-up meetings)	4%	24	4%	25	4%	22	4%	22	5%	28
Helping (helping, managing or mentoring people)	5%	26	5%	27	5%	25	5%	26	5%	28
Networking (maintaining relationships)	2%	10	2%	9	2%	12	2%	11	2%	10
Other Activities										
Learning (honing skills, continuous learning, trainings)	3%	17	3%	14	4%	22	3%	19	3%	16
Administrative tasks	2%	12	2%	11	3%	14	2%	11	3%	15
Breaks (bio break, lunch break)	8%	44	8%	44	8%	45	8%	44	8%	45
Various (*e.g.*, traveling, planning, infrastructure set-up)	3%	21	3%	17	5%	27	3%	19	4%	25
Total	9.08 hours		9.12 hours		9.05 hours		9.17 hours		9.15 hours	

The left number in a cell indicates the average relative time spent (in percent) and the right number in a cell the absolute average time spent (in minutes).

Advantages of Research

- Huge sample size (5,000+ responses)
- Thorough statistical analysis on data collected through survey
- Implications for optimizing developer workdays, evaluating success, and measuring productivity in SE work.
    - *How do we make good days typical?*
    - *Productivity not just defined by code!*

Limitations

- Only conducted at Microsoft
  - *What about developers at other companies?*
- “Goodness” and “badness” are relative
- The typicality and goodness of workdays are not binary
- Difficult for software engineer to self-report how much of your day was spent doing specific tasks

Relevance to class

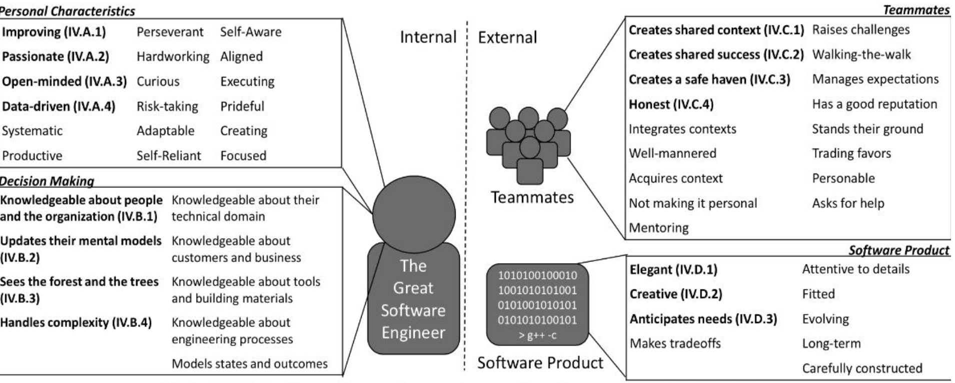
- **Relevance to this class:** Software engineering tasks and artifacts (work products) correlate with good and bad days for developers.
  - Ex) On average, good workdays have less documentation, requirements analysis, etc.

Activity: 1-2-4-All

- Take a few minutes to individually think about the discussion topic. [1]
  - Write down your thoughts and ideas.
- Find a partner in class to discuss your thoughts with, come up with four main points. [2]
- Find another set of partners to discuss your thoughts, agree on two main points. [4]
- Share with the class [All]
  - Student with the closest birthday is the presenter 🎂

Activity: 1-2-4-All

*How can SE courses (like this one) be better designed and structured to help students become **great software engineers** and have typically **good workdays** in their careers?*



Personal Characteristics

Improving (IV.A.1)	Perseverant	Self-Aware
Passionate (IV.A.2)	Hardworking	Aligned
Open-minded (IV.A.3)	Curious	Executing
Data-driven (IV.A.4)	Risk-taking	Prideful
Systematic	Adaptable	Creating
Productive	Self-Reliant	Focused

Decision Making

Knowledgeable about people and the organization (IV.B.1)	Knowledgeable about their technical domain
Updates their mental models (IV.B.2)	Knowledgeable about customers and business
Sees the forest and the trees (IV.B.3)	Knowledgeable about tools and building materials
Handles complexity (IV.B.4)	Knowledgeable about engineering processes
	Models states and outcomes

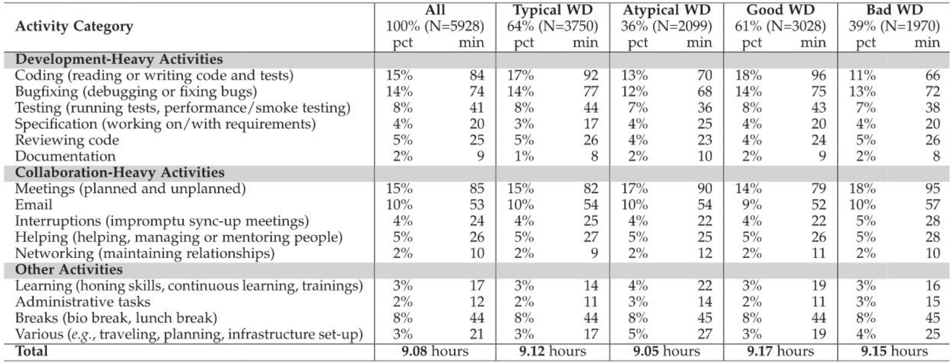
Teammates

Creates shared context (IV.C.1)	Raises challenges
Creates shared success (IV.C.2)	Walking-the-walk
Creates a safe haven (IV.C.3)	Manages expectations
Honest (IV.C.4)	Has a good reputation
Integrates contexts	Stands their ground
Well-mannered	Trading favors
Acquires context	Personable
Not making it personal	Asks for help
Mentoring	

Software Product

Elegant (IV.D.1)	Attentive to details
Creative (IV.D.2)	Fitted
Anticipates needs (IV.D.3)	Evolving
Makes tradeoffs	Long-term
	Carefully constructed

Fig. 1. Model of attributes of great software engineers, with attributes we discuss in detailed in bold.



Activity Category	All 100% (N=5928)		Typical WD 64% (N=3750)		Atypical WD 36% (N=2099)		Good WD 61% (N=3028)		Bad WD 39% (N=1970)	
	pct	min	pct	min	pct	min	pct	min	pct	min
Development-Heavy Activities										
Coding (reading or writing code and tests)	15%	84	17%	92	13%	70	18%	96	11%	66
Bugfixing (debugging or fixing bugs)	14%	74	14%	77	12%	68	14%	75	13%	72
Testing (running tests, performance/smoke testing)	8%	41	8%	44	7%	36	8%	43	7%	38
Specification (working on/with requirements)	4%	20	3%	17	4%	25	4%	20	4%	20
Reviewing code	5%	25	5%	26	4%	23	4%	24	5%	26
Documentation	2%	9	1%	8	2%	10	2%	9	2%	8
Collaboration-Heavy Activities										
Meetings (planned and unplanned)	15%	85	15%	82	17%	90	14%	79	18%	95
Email	10%	53	10%	54	10%	54	9%	52	10%	57
Interruptions (impromptu sync-up meetings)	4%	24	4%	25	4%	22	4%	22	5%	28
Helping (helping, managing or mentoring people)	5%	26	5%	27	5%	25	5%	26	5%	28
Networking (maintaining relationships)	2%	10	2%	9	2%	12	2%	11	2%	10
Other Activities										
Learning (honing skills, continuous learning, trainings)	3%	17	3%	14	4%	22	3%	19	3%	16
Administrative tasks	2%	12	2%	11	3%	14	2%	11	3%	15
Breaks (bio break, lunch break)	8%	44	8%	44	8%	45	8%	44	8%	45
Various (*e.g.*, traveling, planning, infrastructure set-up)	3%	21	3%	17	5%	27	3%	19	4%	25
Total	**9.08** hours		**9.12** hours		**9.05** hours		**9.17** hours		**9.15** hours	

The left number in a cell indicates the average relative time spent (in percent) and the right number in a cell the absolute average time spent (in minutes).

Activity: Fish Bowl Panel [*example*]

- Rotating panel with or without moderator
- One empty seat for audience member to fill in the front of the room
  - Allows anyone from various backgrounds to participate and contribute to discussion
- Interactive discussion on topic or question of choice:
  - i.e. *what makes a great software engineer?*

Next Class…

- No class Monday (9/11)
  - Go to the career fair
- HW1 (due tonight at 11:59pm)

- PM0 (due Monday 9/11 at 11:59pm)
- PM1 (due Friday 9/22 at 11:59pm)
- Requirements Analysis next week (9/13)