Sensing and Supporting Developer Productivity

CPSC 539F

Thomas Fritz
University of British Columbia

Objectives for today

- Introduction Me & Your
- Course Overview
- Research Projects
- Next steps

Introduction

Boosting Developer Productivity

Understanding developer productivity

- Examine productivity perceptions of individuals & teams
- Identify productive behavior & impediments to productivity

Sensing developers' productivity

- Identify measures of productivity, focus, and task difficulty
- Examine use of biometric and computer interaction sensors

Supporting productive behavior

- Provide awareness & actionable insights
- Reduce costly interruptions
- Prevent bugs / defects

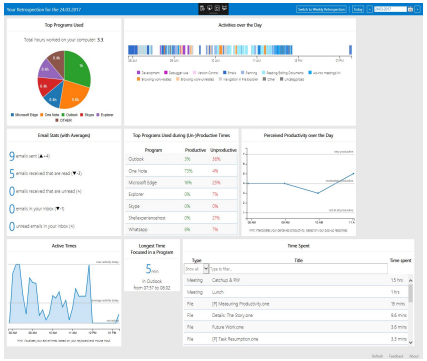
Developer Productivity

Unde

Sensi

Supporting

Developer Analytics



Sensing code difficulty

FlowLight reducing interruptions

Developer Analytics & Retrospection

What does it mean for developers to be productive?

Survey

379 developers

28 questions

Observations

11 developers

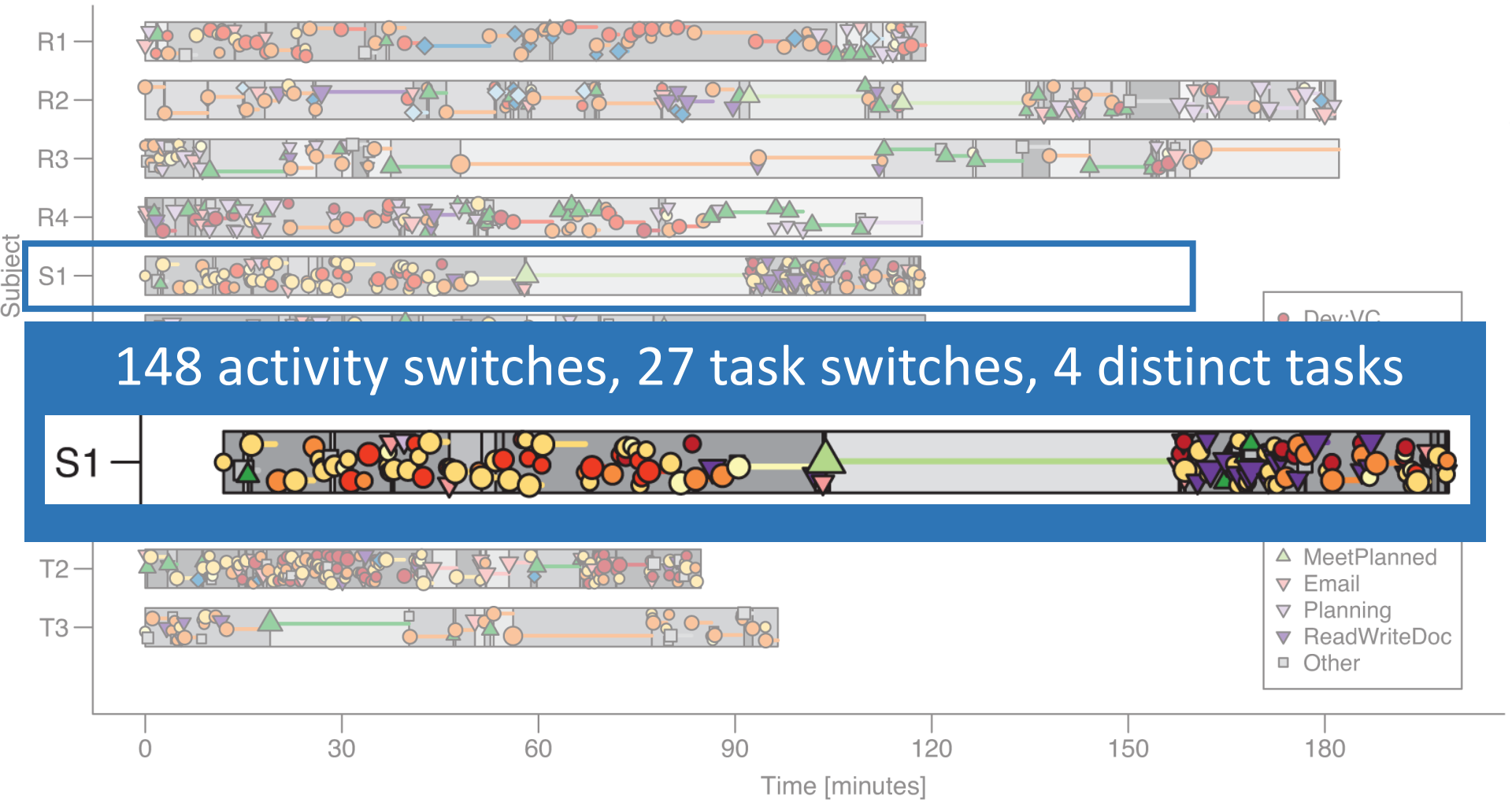
4 hours, 2650 events

Monitoring

20 developers

220 days, 1350 ratings

Observed work flow



Observed work flow

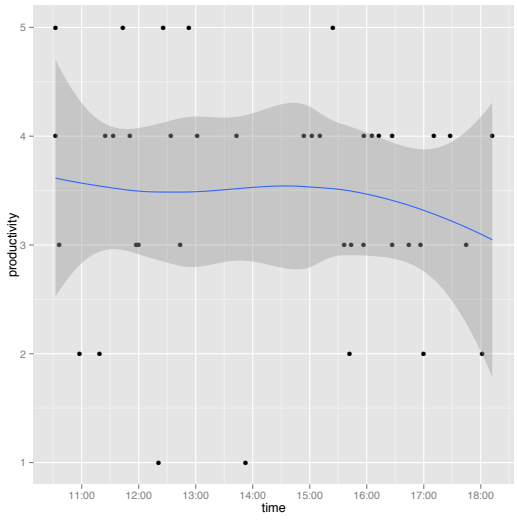
Participants felt fairly productive, yet switched frequently
between tasks (13.5 times/hour) and
between activities (47 times/hour)

Context-switches generally reduce productivity, but costs vary

"[To] stop and work on a different task is a more costly context switch than writing a quick email." (S1)

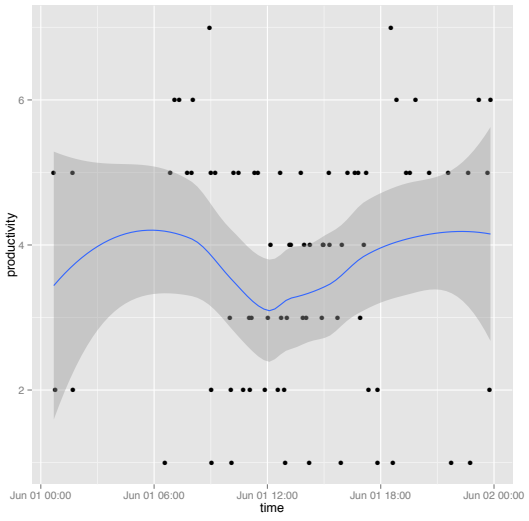
[TSE'17]

Productive times



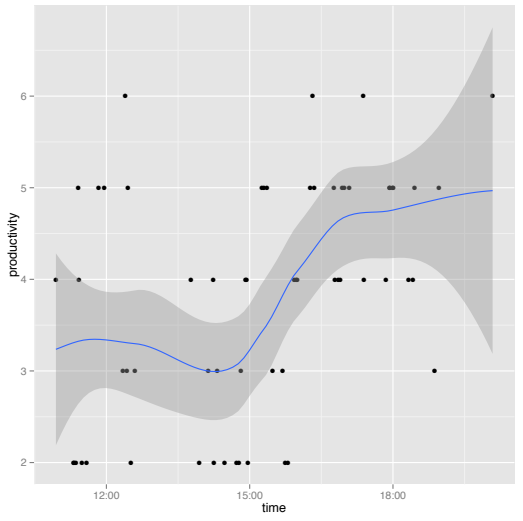
Morning person

(20%)



Low at lunch

(35%)



Afternoon person

(40%)

Developers' perceived productivity follows **habitual patterns**

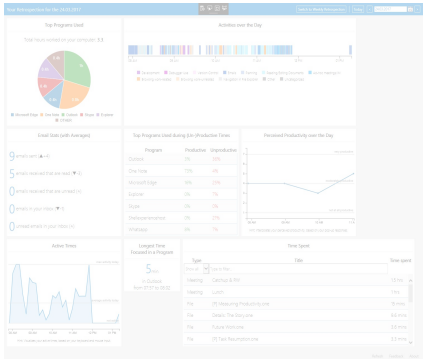
Developer Productivity

Unde

Sensi

Supporting

Developer Analytics



Sensing code difficulty

FlowLight
reducing interruptions

Biometric Sensing of Code Difficulty

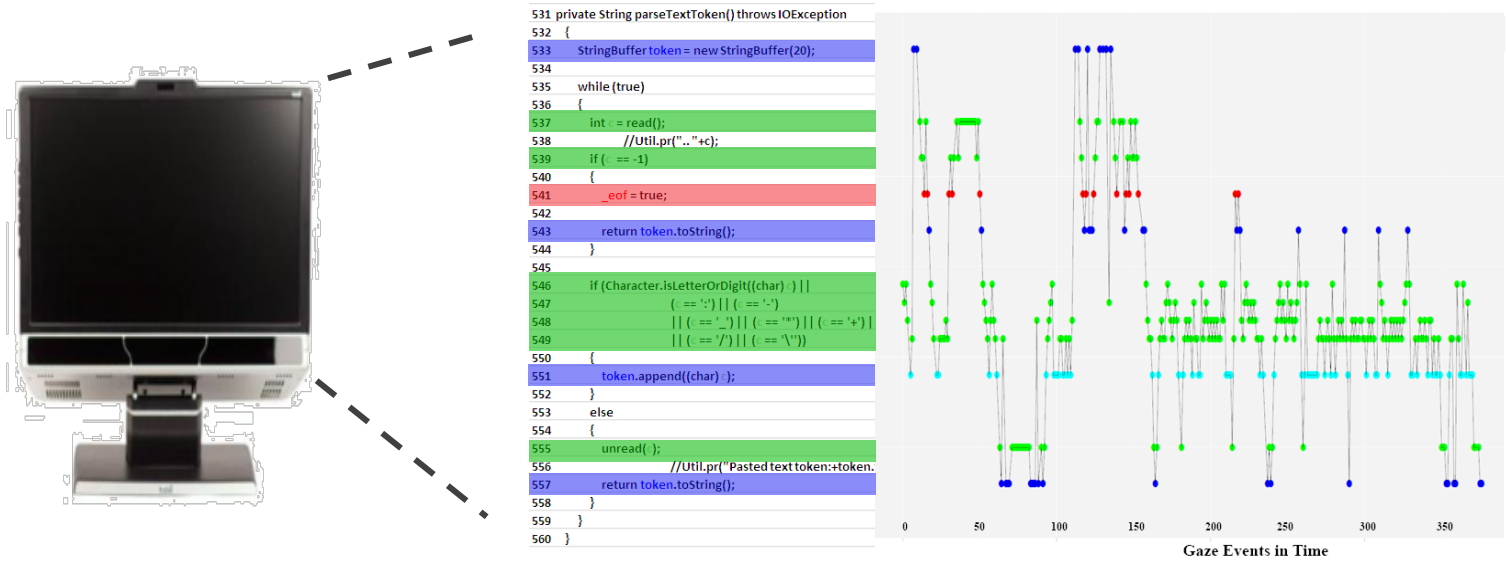
Lab & field studies

Biometric sensors can be used to predict code difficulty and quality concerns

→ **Prevent bugs**

Tracing Software Developers' Eyes for Change Tasks

Understanding developers' code interactions for better tool support



Study with 12 professional developers and 10 students

- Developers only look at small fragments of code elements and often follow data flow within a method

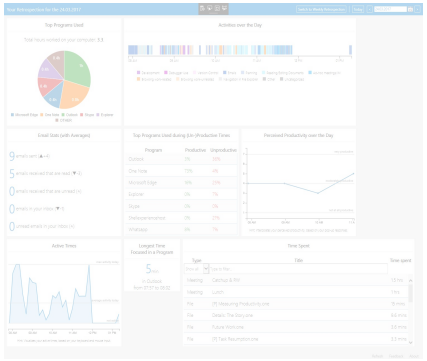
Developer Productivity

Unde

Sensi

Supporting

Developer Analytics



Sensing code difficulty

FlowLight
reducing interruptions

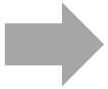
FlowLight – Reducing Costly Interruptions

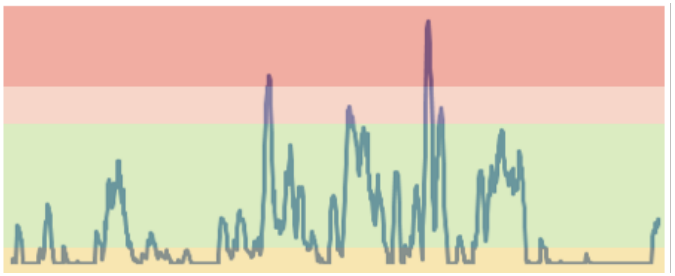
PULSATING RED

RED

GREEN

YELLOW

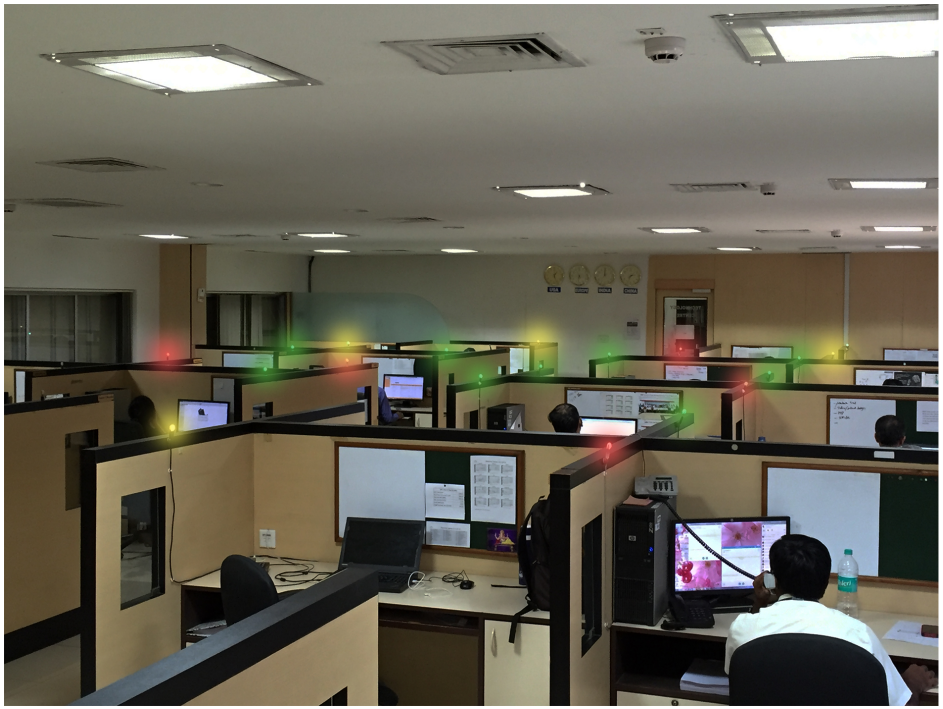




Individualized Thresholds

FlowLight – Reducing Costly Interruptions

Field study with 449 participants, 12 countries

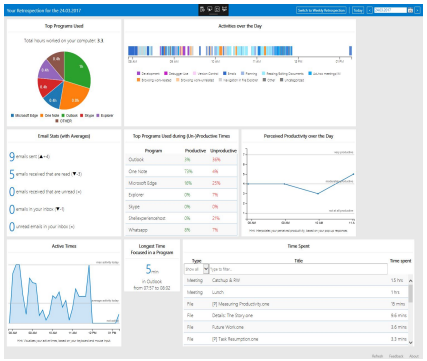


49% less interruptions
85% continued using it on a daily basis

BBC WORLD NEWS
THE NEW YORKER
New Scientist
THE TIMES
WSJ
cnBeta.COM 中文业界资讯站
NRJ
GeekWire
DIGITAL TRENDS
The Telegraph
NBC NEWS

Boosting Developer Productivity

Understanding developer productivity



Sensing developers' productivity

Supporting productive behavior

- Provide awareness & actionable insights
- Reduce costly interruptions
- Prevent bugs / defects
- **Provide easy access to relevant information**

Introduce Yourself!

Who are you?

What are you interested in?

What would you like to learn about?

Course Overview

Course Objectives

- Deepen and broaden your knowledge of Software Engineering research (and some HCI) by ***reading, reflecting and discussing*** current and classic literature

- Learn to ***define and study research questions***

- Experience a glimpse of Software Engineering research through a research ***project*** work

Focus on Research and the Process

The ***process*** is important

- Identifying interesting research questions (RQs)
- determining how to investigate them and running analysis
- presenting and writing up results

Research is mostly an ***iterative*** process

- Identifying relevant RQs is difficult and discussing and then revising them is important and part of research

Course Format

Outside of class

- 2 to 4 papers per week (more papers in the beginning, to get into domain, less papers & more meetings/activities later on)

  read papers, think about them, write a short response paper

- Research Project

  find a research question, write and present a proposal, do it, write it up and present it (you can team up for it)

In class

- Discussion and moderation of research papers

  student moderator gives a 10min overview, leads discussion, everyone else participates in discussion

- Other times: small activities, weekly meetings on project progress, presentation

Tentative Schedule

Sept 6 Course overview

Sept 11 Research in Software Engineering (Papers & Discussion P&D)

- Preliminary guidelines for empirical research in software engineering
- Experimental models for validating technology

Sept 13 Research in SE & Dev Productivity (P&D & Activity)

- What makes good research in SE?
- Software Developers' Perceptions of Productivity.

Sep 18 Productivity (P&D)

- Using a defined and measured personal software process.
- Bored Mondays and Focused Afternoons: The Rhythm of Attention and Online Activity in the Workplace

Tentative Schedule (2)

Sept 20 Sensing Developers & Cognitive Load (P&D)

Psycho-physiological measures for assessing cognitive load

Sept 25 Eye-Tracking and Support (P&D & Activity)

EyeDE: Gaze-enhanced Software Development Environments

Analyzing Individual Performance of Source Code Reviewers' Eye Movement

Sep 27 Project Proposal

Draft due + meetings

Oct 2 Project Proposal Presentation

presentation + write up

Tentative Schedule (3)

… (more to come)

Nov 15 Project Meetings

short scrum + discuss findings and results

Nov 20 & 22 Project Presentations

final report due + presentations

Nov 27 & 29 Program Committee Meetings

report reviews due + in class discussion of reports/results

Focus & Topics of Course

- (Developer) Productivity
- Biometric sensing
- Developer activities, work fragmentation, interruptions
- Data on developers: interaction logs, biometric data, observation logs, activity logs …
- Developer support
- Self-monitoring and goal setting
- Program comprehension, software evolution, …
- Empirical Research and studies of software developers (quantitative and qualitative)

Grading

- 55% Project (including proposal, report, presentation)
- 10% Peer reviews of project reports
- 35% Readings (including response papers, moderation, class participation)

Response Papers

- Encouragement to read and reflect
  - Class discussions work better if everyone has read and thought about the paper
- At most one page per class (300 to 500 words)
- **NOT a summary**. Think of it this way
  - If I asked you what you thought about a movie you recently went to, you wouldn't just summarize it
- Grading based on "thoughtfulness"
- Due by **8pm** on day before class
  - Submit by email

Response Papers

- Questions of interest
  - What did you think about it and what did you find important or interesting?
  - What are main contributions of the paper?
  - What are strengths or weaknesses of the paper/research?
  - What are five questions you have about it?
  - What could be improved?
  - How could you imagine extending the work?
  - Do you agree or disagree with the findings?
  - How does the research relate to other papers for this lecture?
  - …
- Express your perspective, ***address all readings*** and ***draw connections between readings*** when possible
- Example provided on web site!

Discussions

- Discuss the research:

  which problem are they trying to address, how are they tackling the problem, how do they evaluate their approach, …

- Share your opinions, ideas and thoughts
- Ask questions about the work
- See what others thought
- Listen and speak actively
- Look for contributions not just flaws in reading

Leading Discussions

- For about 1 to 2 papers (**depends** on class size)
- Prepare for discussion
  - Prepare introduction/presentation (**at most 10 to 15mins**)
  - Provide additional background on subject
  - Prepare interesting and challenging points and questions for discussion
  - Think about overall structure, how to make it interactive and how to keep discussion going
- Hand in a brief (one paragraph) description of your plan
- Sign-up at the end of next class

Moderation

- Available time approx. 40mins; keep time in mind
- Introduce and provide background (10mins)
- Moderate an interactive discussion and include other student's perspectives and ideas
- Try to involve all participants
- Give impulses for discussion
- If an interesting topic comes up, be flexible about veering from your discussion plan
- Be respectful

Research Projects

Research Project – Empirical Analysis

- Identify a real problem developers face / investigate specific aspect of SE
- Read related work and determine your niche
- Identify relevant/interesting research question
- Determine how to address the research question
- Run analysis
- Write up results in a scientific manner

Some Possible Projects

- Hands-on project with biometric sensors

- Examine developer activity and productivity
- Analyze biometric / eye-tracking or interaction data
- Examine software repository histories and metrics

Research Project – Empirical Analysis

- Each project accompanied by a paper (max. 5 or 10 pages)
- Individual or in groups (up to 3 people, depending on class size)
- One page project proposal *draft* due on **September 26th**
- Project proposal presentation
- Final one- to two-page proposal due on **October 2nd**
- Written report due on **November 19th**
- Project presentation

Research Project – Empirical Analysis (3)

- Project report: ACM paper format
- One-on-one meetings shortly before and after project proposal is due
- SCRUM: class meetings, each one has 2mins to state what they have done last week and what they will do next week
- Continuous short progress meetings (depending on class size)
  - discuss progress, next steps, open questions, keep on track …
  - take advantage of them, i.e. prepare and ask!
  - 10 to 20 minutes

Peer Review of two Project Reports

- Research communities rely on peer reviews of results, if you want to be a researcher you need to learn about critiquing research papers
- Paper review will also help you to learn what is important when you write up your own work
- Templates will be provided

  Summary, what are its strengths, what are the weaknesses

  (realize that the authors would usually read it, so be constructive)

Peer Review (2)

- Assess projects like a program committee
  - Everyone will read and review two project reports
  - Reviews are organized via OLAT
- Hand in review (will also be sent to authors)

Some More & Next Steps

Me

- I’m here to help
- Talk to me if you wand feedback or need help
- Talk to me if you do not find a topic or want to discuss your idea
- In class, I’m here to discuss

To Dos

- Choose papers by end of next class that you would like to present
- Start thinking about projects as soon as possible: what are you interested in?

Next week Monday

- Two papers on Empirical Research
  - **Experimental Models for Validating Technology.** Zelkowitz et al. IEEE Computer, 1988.
  - **Preliminary guidelines for empirical research in software engineering,** Kitchenham et al. TSE'02.
- Read and write short response paper
- Submit by email

More Information

- See website:
- https://github.com/ubccpsc/ssdp/tree/2017sep/

- Contact:
  - Thomas Fritz fritz@cs.ubc.ca