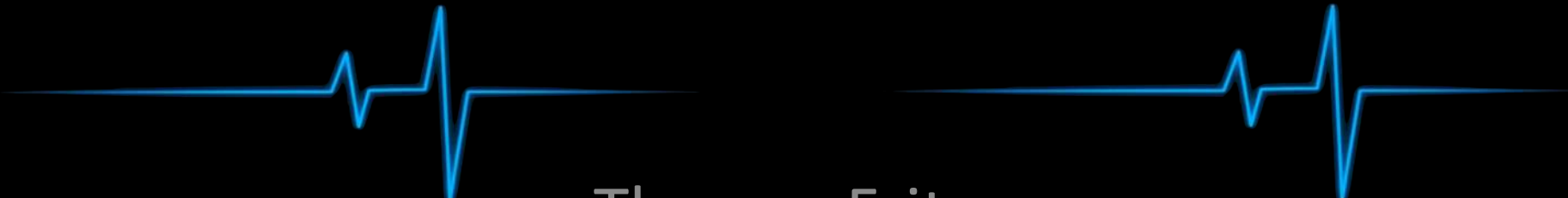


Human Aspects of Software Engineering



Thomas Fritz
University of Zurich



University of
Zurich^{UZH}

Objectives for today

- Introduction Me & Research Overview
- Introduction You
- Course Overview
- Research Projects
- Next Steps
- Exercise

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

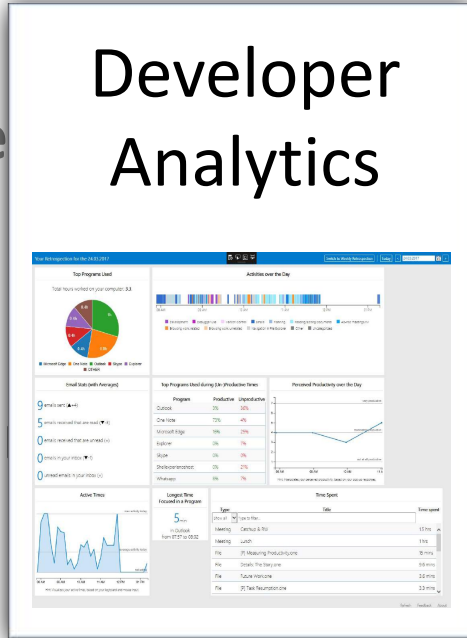
Supporting productive behavior

- Provide awareness & actionable insights
- Reduce costly interruptions & support focused work
- Prevent bugs / defects
- (Semi-)Automate Workflows

Developer Productivity

Under

Developer Analytics



Sensi

Sensing code difficulty



Supporting

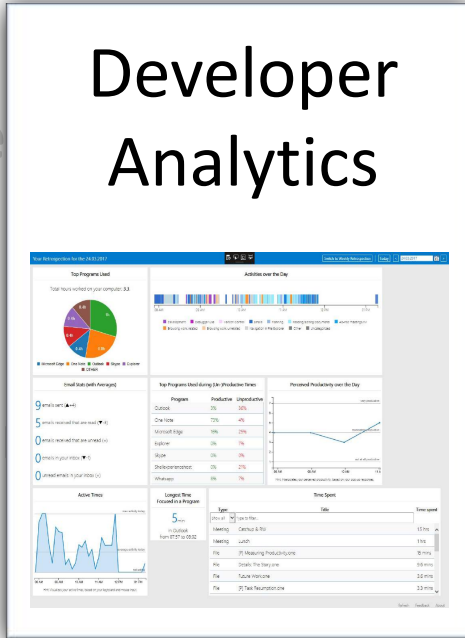
FlowLight reducing interruptions



Developer Productivity

Under

Developer Analytics



Sensi

Sensing code difficulty



Supporting

FlowLight
reducing
interruptions





Developer Productivity & Analytics

What does it mean for developers to be productive?



Survey

379 developers
28 questions



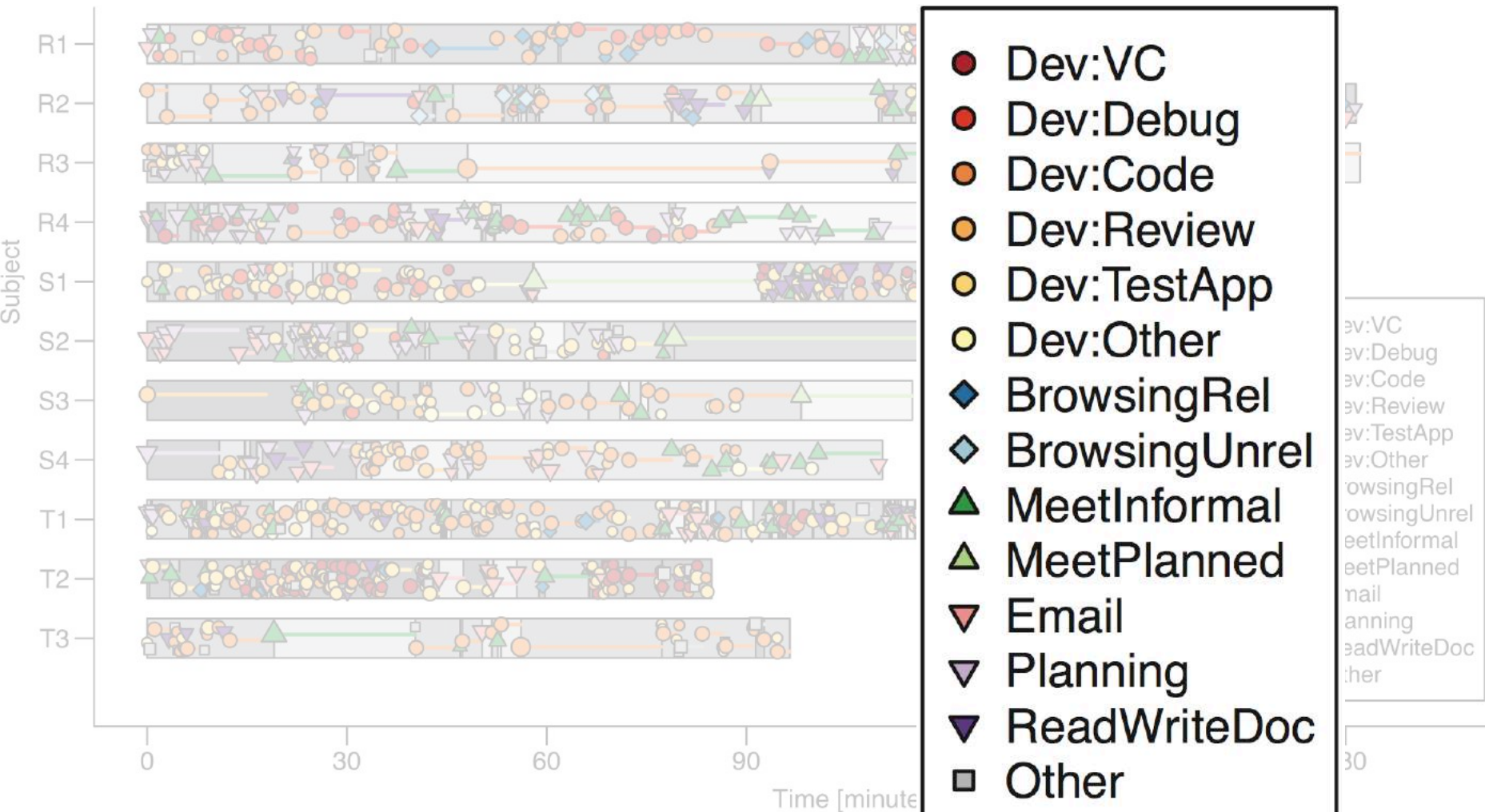
Observations

11 developers
2x2 hours, 2650 events

Developers feel productive when they make **progress on tasks** with **few context switches / interruptions**

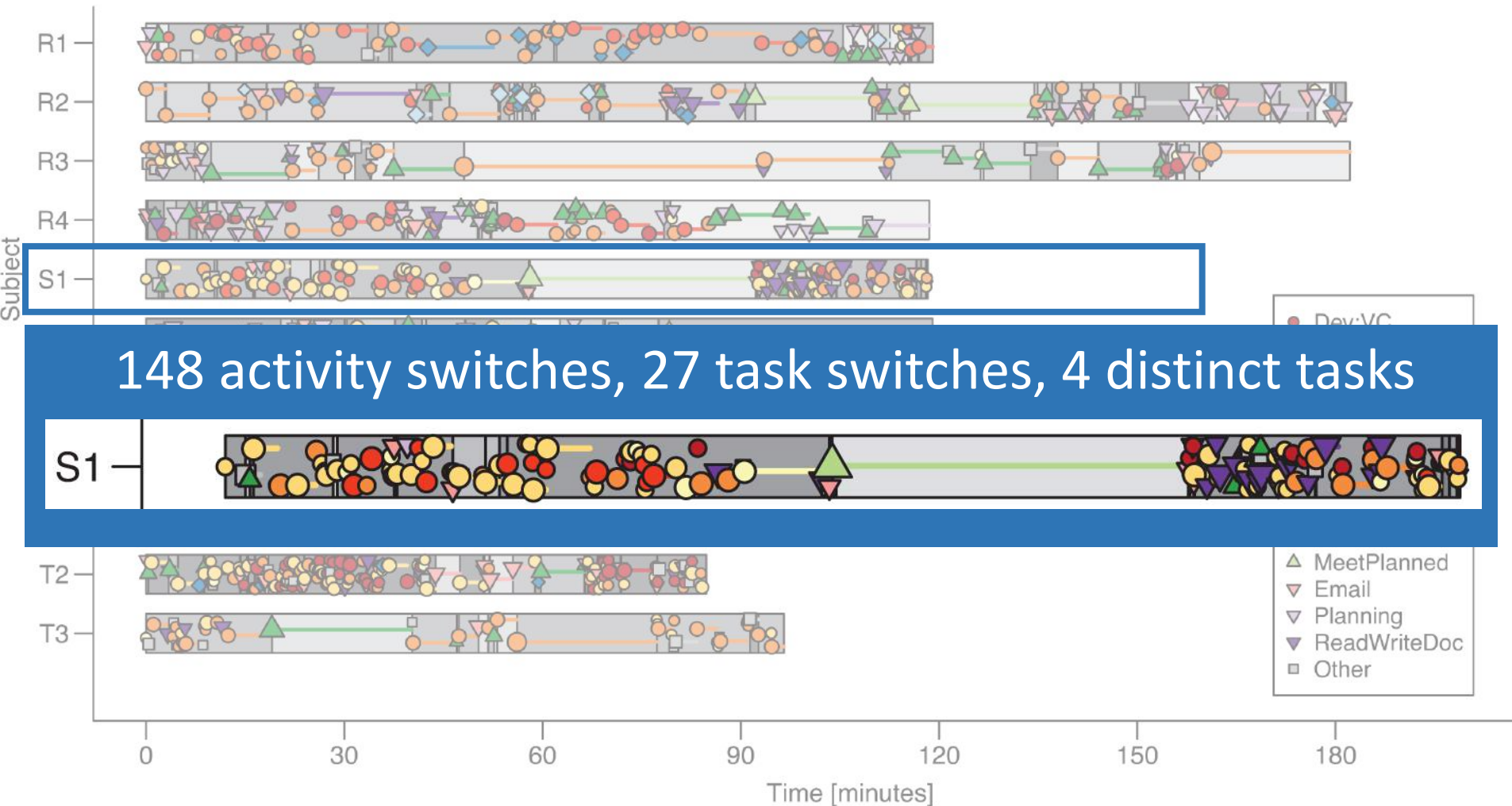


Observed work flow



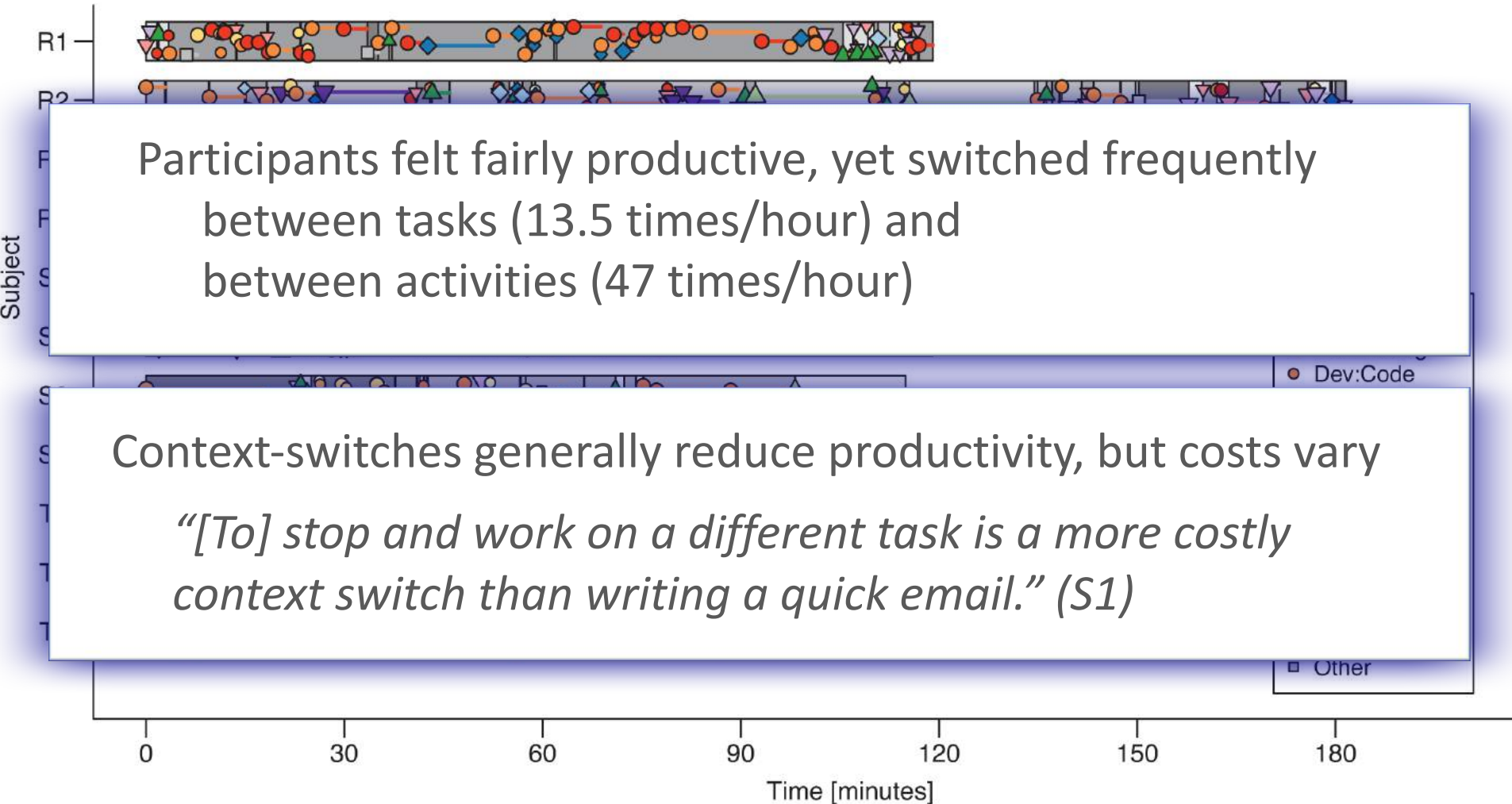


Observed work flow





Observed work flow



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

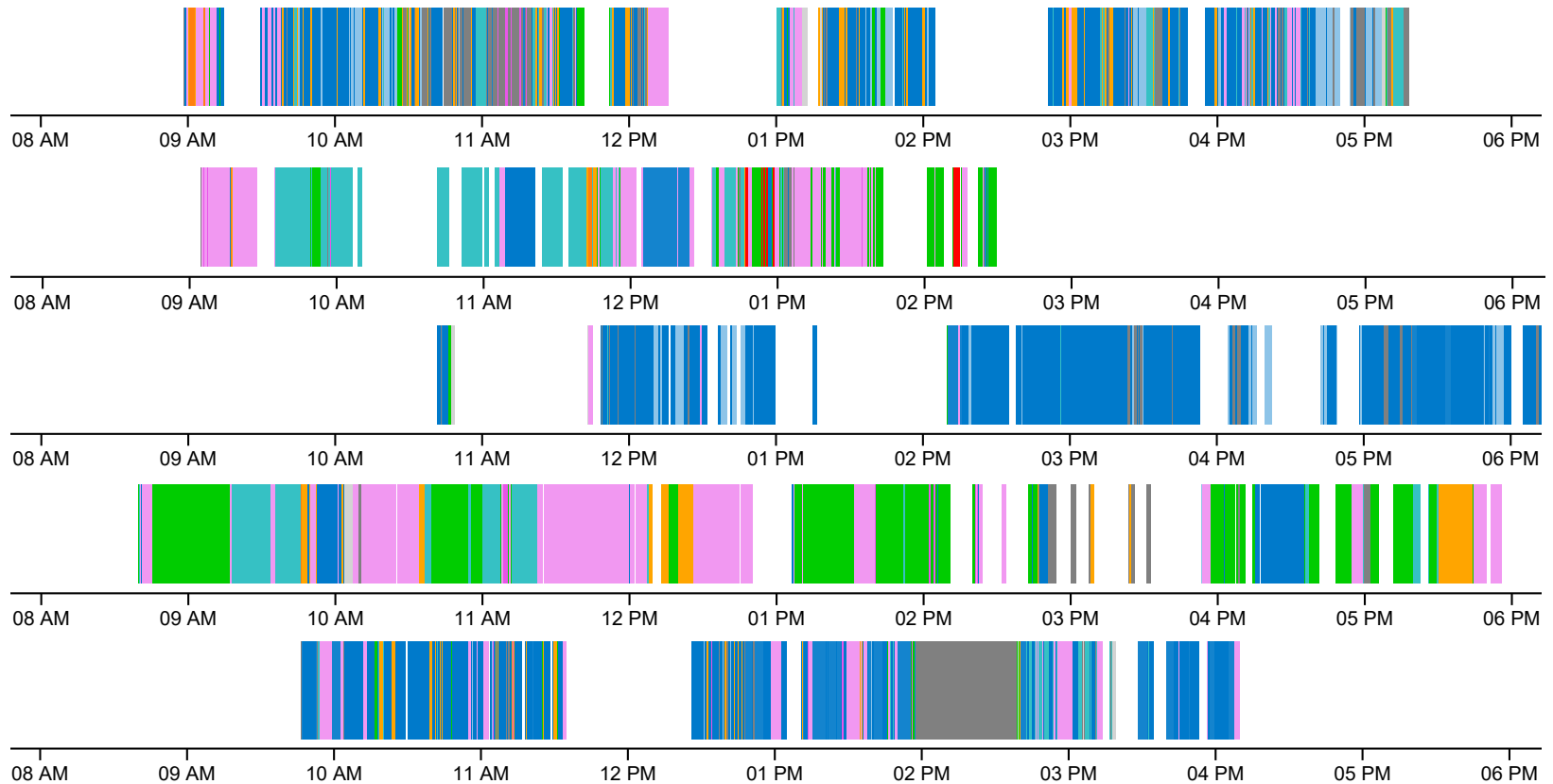
20 developers
220 days, 1350 ratings

Developers feel productive when they
make **progress on tasks** with
few expensive context switches / interruptions



Monitored activities

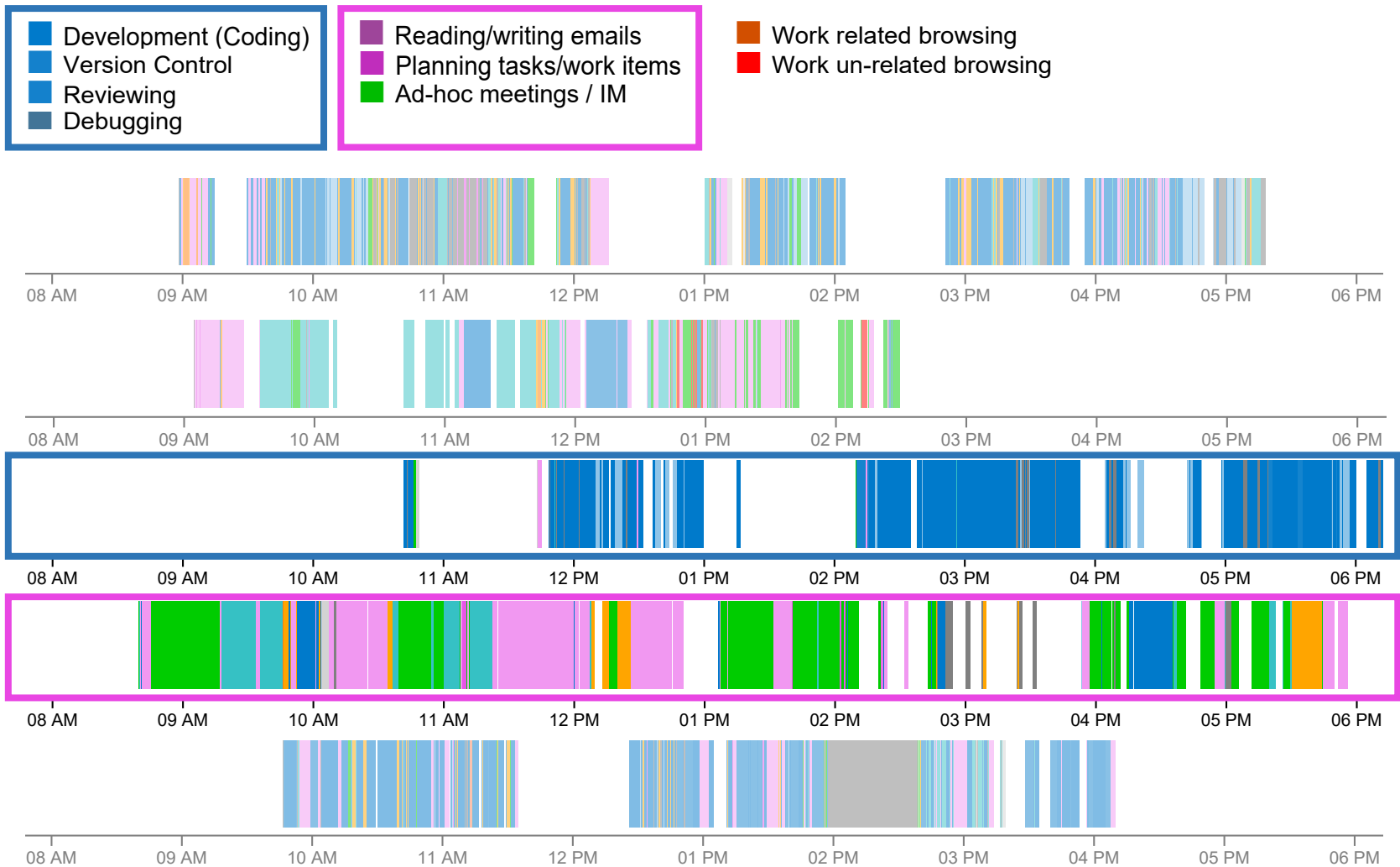
- Development (Coding)
- Version Control
- Reviewing
- Debugging
- Reading/writing emails
- Planning tasks/work items
- Ad-hoc meetings / IM
- Work related browsing
- Work un-related browsing



Development work is **highly fragmented**



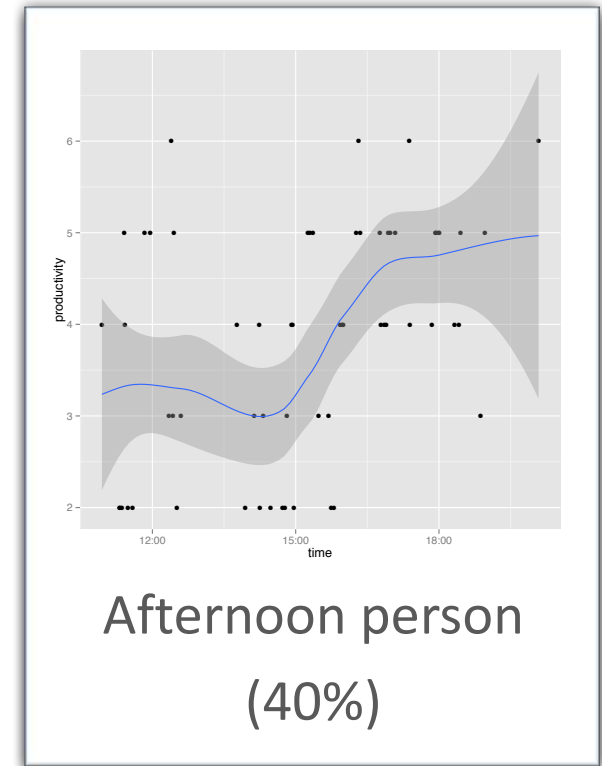
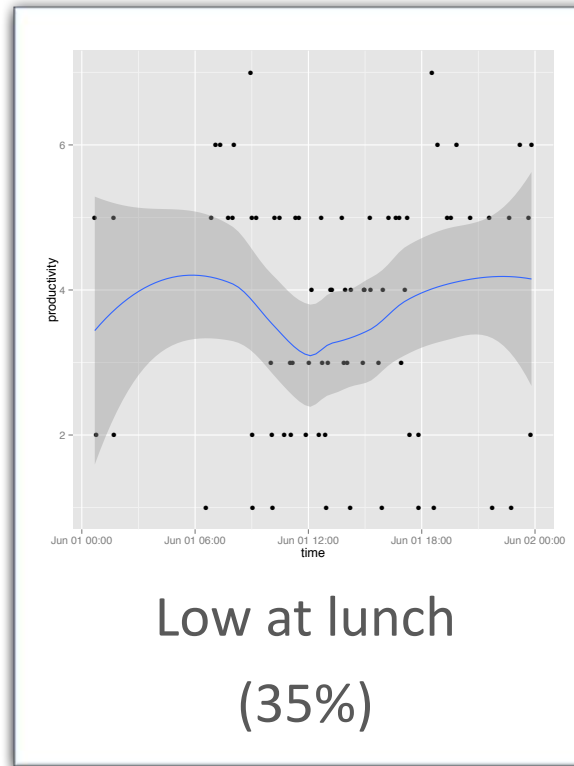
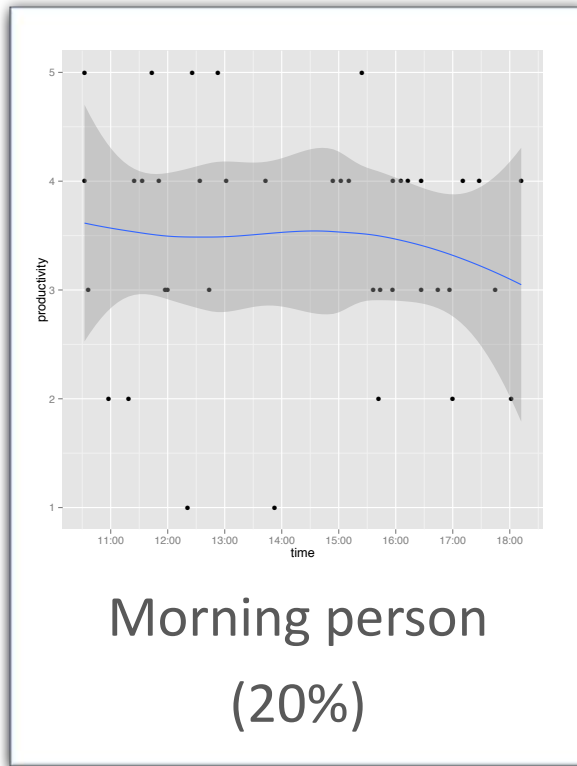
Monitored activities



Development work is highly fragmented and **follows themes**

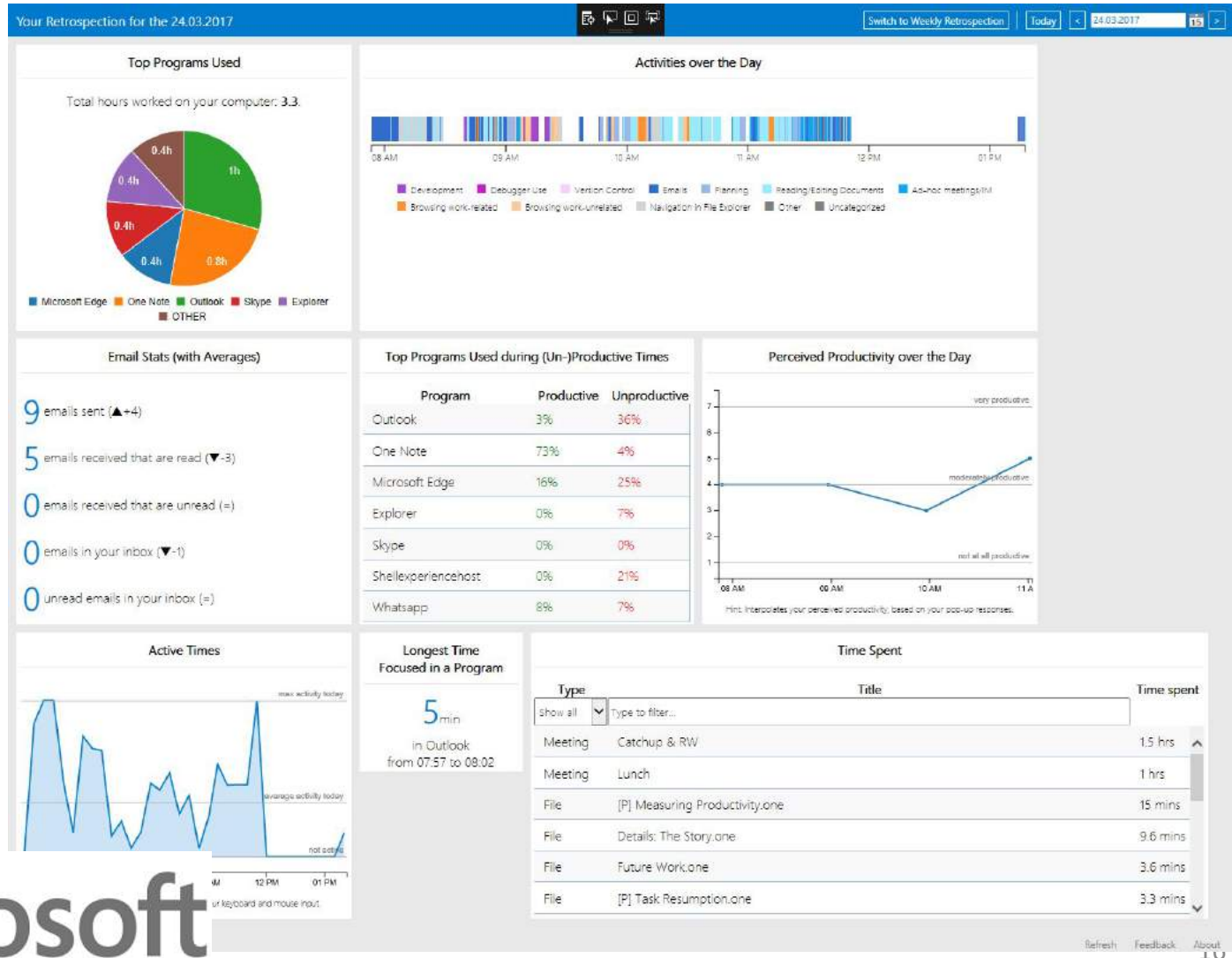


Productive times



Developers' perceived productivity follows
habitual patterns

Developer Retrospection (Fitbit for Developers)



Microsoft

Developer Productivity

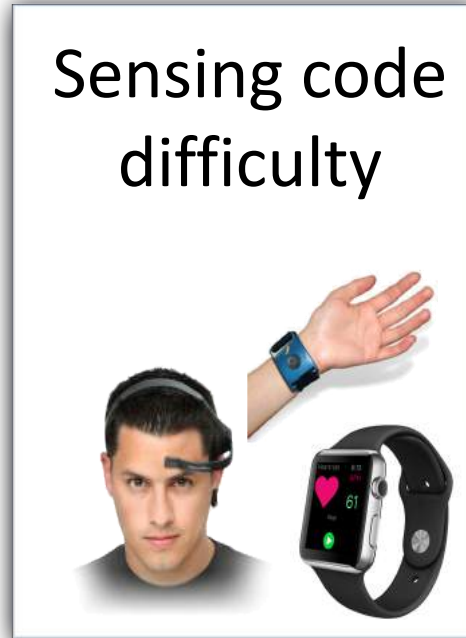
Under

Developer Analytics



Sensi

Sensing code difficulty



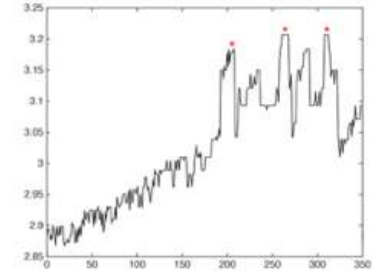
Supporting

FlowLight reducing interruptions





Psycho-physiology (biometrics)



Psychological (*mind*)

- Mental load
- Task engagement
- Excitement
- Emotions
- ...



Physiological (*body*)

- Brainwaves
- Sweat
- Heart rate variability
- Pupil size
- Eye blink rate
- ...



Biometric Sensing of Code Difficulty



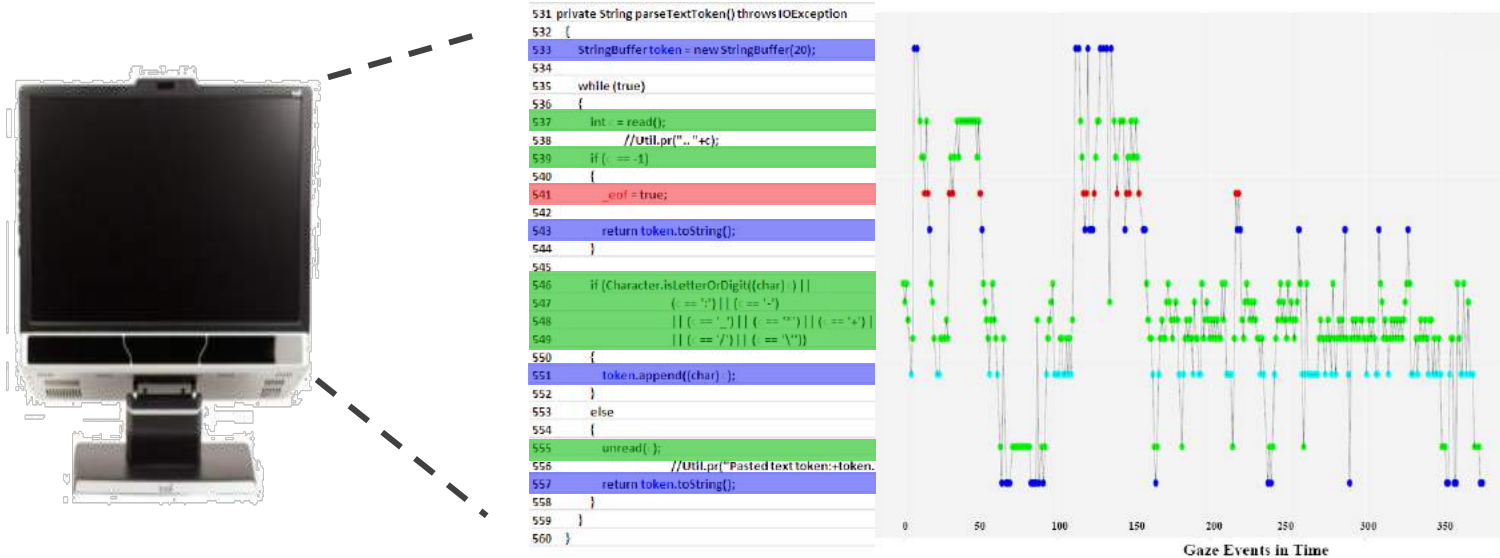
Lab & field studies



Biometric sensors can be used to predict code interruptibility, difficulty and quality concerns
→ **Prevent bugs & costly interruptions**

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

Under

Developer Analytics



Sensi

Sensing code difficulty



Supporting

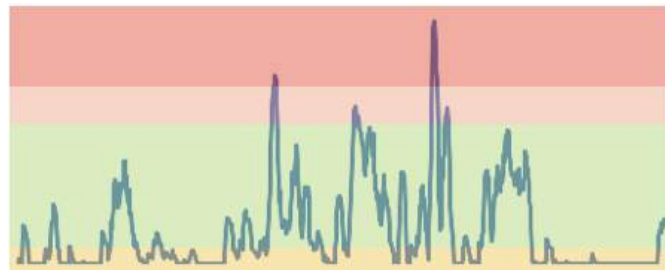
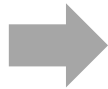
FlowLight reducing interruptions



FlowLight – Reducing Costly Interruptions



PULSATING RED
RED
GREEN
YELLOW

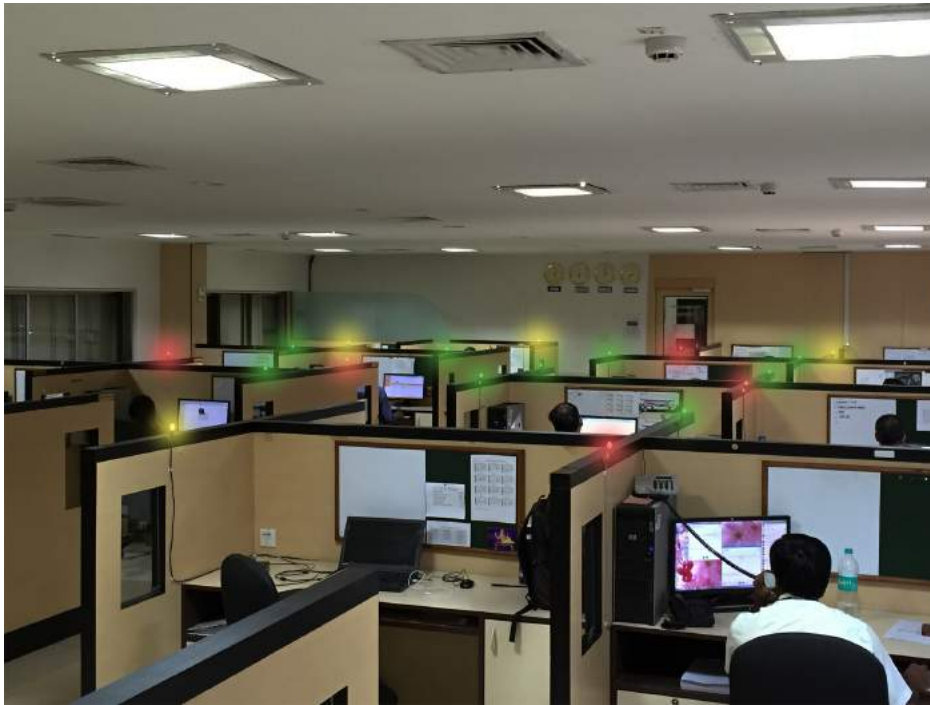


Individualized Thresholds



FlowLight – Reducing Costly Interruptions

Field study with 449 participants, 12 countries



46% less interruptions
85% continued using it
on a daily basis

BBC
WORLD
NEWS

THE
NEW YORKER

New
Scientist

THE  TIMES

WSJ

GeekWire

 DIGITAL TRENDS

The Telegraph

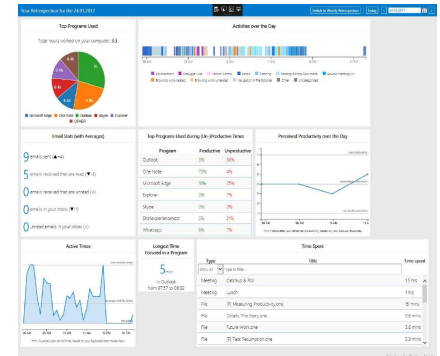
 cnBeta.com
中文业界资讯站


NBC NEWS



Boosting Productivity & Well-being

Understanding developer productivity



Sensing developers' productivity



Supporting productive behavior

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



Our Research

Boosting (developer) Productivity



Developer Productivity
& Analytics



Sensing Developers



Monitoring &
Empowering
Productivity

Mental Health
& Safety

Increasing Focus

Workflow
Automation



Microsoft



iQmetrix



Please, 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

Non-Traditional Course Format

Outside of class

- 2 to 3 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
One student introduces paper (5min), everyone participates in discussion
- Other times: small activities, weekly meetings on project progress, presentation

No exams, however, projects will take some time. [6 ECTS]

Tentative Schedule

Sept 17 Course overview & Introduction

Sept 24 (Empirical) Research in Software Engineering ([Papers & Discussion](#))

What makes good research in software engineering?

A practical guide to controlled experiments of software engineering tools...

Oct 1 Developer Productivity & Support ([P&D](#))

The Work Life of Developers: Activities, Switches and Perceived Productivity

Code Bubbles: Rethinking the User Interface Paradigm of IDEs

Evaluating Cues for Resuming Interrupted Programming Tasks

Oct 8 Sensing and Supporting Code Difficulty

Learning a Metric for Code Readability

Using Bio(Metrics) to Predict Code Quality Online

Helping developers help themselves: automatic decomposition of code review changesets

Tentative Schedule (2)

Oct 15/16 Proposal discussions (One-on-one meetings; writeup due before)
→ sign up early by email to fritz@ifi.uzh.ch with 3 preferences of 30mins slots for Mon / Tue

Oct 22 Proposal Presentations (presentation)

Oct 29 SCRUM in Research & Eye-Tracking in SE & ML

Improving Automated Source Code Summarization via an Eye-Tracking Study of Programmers

An eye-tracking study on the role of scan time in finding source code defects.

A brief introduction to Machine Learning

Nov 5 Written progress update

brief write up of progress

Tentative Schedule (3)

Nov 12 SCRUM & Mood / Emotions in SE (P&D)

Do moods affect programmers' debug performance? ...

Nov 19 - Dec 9 Weekly Meetings

Dec 10 Project Report due

Dec 14 Peer-reviews due

Dec 17 Project Presentations in class

final report due + presentations

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

- 65% Project (including proposal, report, presentation)
- 25% Readings (including response papers, class participation & leading discussion)
- 10% Peer reviews of project reports

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
- **Sometimes**, short question to be addressed.
- 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
- **If it's your turn:** introduce the paper for ~5mins & come up with questions,
- **SIGN UP NOW**

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 2 people, depending on class size)
- One page project proposal *draft* due on **October 14**
- Project proposal presentation
- Final one- to two-page proposal due on **October 24**
- Written report due on **December 9**
- Project presentation

Research Project – Empirical Analysis (2)

- 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 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 want 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 that you would like to introduce/present **by end of today**
- Start thinking about projects as soon as possible: what are you interested in?

Next week Monday

- Two papers on Empirical Research
 - **What makes good research in software engineering?** Shaw, Int. Journal of Software Tools for Technology 2002.
 - **A practical guide to controlled experiments of software engineering tools with human participants**, Ko et al., Empirical Software Engineering 2013.
- Read and write short response paper
 - First part, brief summary
 - Second part: Think about a problem you have/had programming / developing software and a way to study it (including what you would examine exactly) or sketch a rough idea of some tool support and how to evaluate it
- Submit by email

Discussion Starter / 3 to 5mins intro

Who is up for the first ones?

- 1) What makes good research in software engineering?
- 2) A practical guide to controlled experiments of software engineering tools with human participants
- 3) The Work Life of Developers: Activities, Switches and Perceived Productivity
- 4) Code Bubbles: Rethinking the User Interface Paradigm of IDEs
- 5) Evaluating Cues for Resuming Interrupted Programming Tasks

More Information

- See website:

<http://www.ifi.uzh.ch/en/seal/teaching/courses/hase.html>

- Contact:

Thomas Fritz

fritz@ifi.uzh.ch

Exercise

In groups, discuss...

- Think about a problem you have/had programming / developing software
- How could you study it?
- How could you improve it?
- How could you show the improvement?

Some problems mentioned...

- Debugging and Java Script
- Loosing sight/overview of whole projects
- Team work: personality and fit
- Finding/reading documentation
- Information overlaod
- How to best gather expertise
- Typing in PL -> productivity
- Leveling expertise
- Library management