The software developer as the knowledge worker of tomorrow

Margaret-Anne (Peggy) Storey
University of Victoria, Canada
Lise Meitner Guest Professor, Lund University 2016-2018

ELLIIT Workshop, Lund, Sweden—April 26, 2017

@margaretstorey
Research goals

Gain an understanding of how developers collaborate to create complex systems
Improve development and communication tools developers rely on

“software is eating the world”
--- Marc Andreesen
“software brings hardware to life”
---Grady Booch
This Car Runs on Code

100-200 million lines of code ~100 processors
70% effort spent on software

> 50% warranty costs due to software
“I know how this was done because I did it”

“I need complete understanding”

Peter Norvig, Coders at Work
Today’s software engineer
“How is this likely done?”
“Can I quickly get an understanding of what I need?”

Peter Norvig, Coders at Work
Engineering at Google

30,000 developers in 40 offices
2 Billion LOC
1 Billion files, 9 million source files
35 million commits, 40,000 commits per day
One monolithic repository!
“Scaling to 1000s of developers — automation is required!” [Jacek Czerwonka]
Modern Developer’s Toolbox

IDE
Source control
Continuous integration, deployment
Code review
Testing frameworks
Issue tracking
Experimentation platforms
Project management
Communication tools -> Social media…
Programming as a "theory building process"

Software is built using tacit knowledge captured in developers’ head(s), and from externalized knowledge embodied in their development tools, channels, and project artifacts [Naur, 1985].
Discussion topics

Theories of media and how media shape software development

How social/communication channels have evolved over time in software engineering

From software development to knowledge work
Discussion topics

Theories of media and how media shape software development

How social/communication channels have evolved over time in software engineering

From software development to knowledge work
The McLuhan Tetrad

The smart phone

Immediate gratification
- App for that
- Connectivity
- Immediate answers
- (Weak) social ties

Avoid phone calls
- Distraction
- Instrument of doom
- Barrier to relationships

Enhance
- What does it amplify?

Reverse
- How does it flip when pushed?

Revive
- (Retrieve)
- What does it bring back?

Obsolesce
- What does it make obsolete?

- Telephone
- Camera
- Scanner
- Desktop software
- MP3 Player

Adapted from “The Laws of Media”, McLuhan & McLuhan, 1988
“If we understand the revolutionary transformations caused by new media, we can anticipate and control them; but if we continue in our self-induced subliminal trance, we will be their slaves.”
[Marshall McLuhan, 1974]
Communities of practice emerge when people share a passion and learn together [Wenger]
Social Media and Participatory Cultures [Jenkins]

Low barriers to artistic expression and engagement

Strong support for sharing one’s creations

Informal mentorship for novices

Members believe their contributions matter

Members care about social connections and what others think about their creations

The Participatory Culture in Software Engineering is not new
Discussion topics

Theories of media and how media shape software development

How social/communication channels have evolved over time in software engineering

From software development to knowledge work
Developer tools over time...
Understanding affordances of channels for communicating knowledge [Wasko et al.]

Communicating knowledge that is embedded in people (F2F, email, chat…)

Communicating knowledge embedded in project artifacts (GitHub, Visual Studio…)

Communicating knowledge embedded in community resources (Books, Usenet)

(new!) Communicating knowledge about social networks (Facebook, Coderwall, Twitter…)

Properties of social dev tools

Architecture of participation
Transparency
Persistence
Emergence of behaviours
How developers **stay up to date** using Twitter

How developers **crowdsource** software documentation

How developers share **tacit knowledge**

How thousands of developers **coordinate which code** is committed and accepted through GitHub

How **bots integrate** collaboration and automation

How **gamification** can enhance **productivity**
Findings:

– Awareness
– Learning
– Relationships
– Strategies
– Why non-adoption

“It was evolving way faster than I was able to keep up with it. And the only way to keep up was to follow some Node.js people on Twitter.”

Crowd Documentation?

Coverage of **API documentation**: 77% of the Java API classes & 87% of Android API classes

**Speed of coverage:**

Code, camera action: How software developers **document and share program knowledge** using YouTube, L. MacLeod, A. Bergen, MA Storey - 2015 23rd IEEE Int. Conf. on Program Comprehension, 2015
“After just seven years on the net, GitHub now boasts almost 9 million registered users. Each month, about 20 million others visit without registering... GitHub is now among the top 100 most popular sites on earth.” [wired.com]
How project owners use pull requests on GitHub for code review

Code quality, style, fit… influence decision to accept

http://www.gousios.gr/blog/How-do-project-owners-use-pull-requests-on-Github
Kevin Kelly, Futurist: “You’ll be paid in the future based on how well you work with robots.”
What is a bot?

A bot is an application that performs automated, repetitive, pre-defined tasks.

Conduit between users and services through a conversational UI.

From setting an alarm, to telling you today’s weather forecast, to gathering information.
Software development Bot roles

**Code** Bots
**Test** Bots
**DevOps** Bots
**Support** Bots
**Documentation** Bots
**Entertainment** Bots

*ChatOps are "putting tools right in the middle of the conversation"
- Jesse Newland, GitHub*

B. Lin, A. Zagalsky, M.-A. Storey, and A. Serebrenik. Why developers are slacking off: Understanding how software teams use slack. CSCW 2016 (poster paper).

Categories also inspired by Sven Peter: [https://svenpet.com/talks/rise-of-the-machines-automate-your-development/](https://svenpet.com/talks/rise-of-the-machines-automate-your-development/)
Gamification: a Game Changer for Managing Technical Debt?  A Design Study

Matthieu Foucault & Margaret-Anne Storey, UVic
Xavier Blanc & Jean Remy Falleri, Universite de Bordeaux, France
Cedric Teyton, ProMyze, Bordeaux, France (under review)
(Distributed) Community formation, awareness, transparency, knowledge curation, motivation, learning, reuse, reputation

Informal processes, geek culture, reliance on search, privacy concerns, fragmentation, interruptions, Vendor lock-in

Programming gurus, end users as developers, verbal discussions, portfolios, competition, communities of practice

In-house expertise/jobs, formal documentation, classroom education, CVs, email lists, need for co-location
Discussion topics

Theories of media and how media shape software development

How social/communication channels have evolved over time in software engineering

From software development to knowledge work
Knowledge workers

Primary job is thinking or creating, often collaboratively
Learning is key

“the most valuable asset of a 21st-century institution, whether business or non-business, will be its knowledge workers and their productivity” --- Peter Drucker, 1957
Knowledge Theory in Software Development

Roles

Developers (stakeholders)

Communities of Practice

Knowledge as Public Good

Knowledge Activities

Tacit Knowledge

fulfill

participate

use / output

create / maintain

Channels / Tools

Knowledge on People & Social Networks

use / output

output / change

Externalized Knowledge

Code

Documentation

Q&A

History

Credit: Alexey Zagalsky
Developer tools from today are the knowledge worker tools of tomorrow:
“GitHub for Education” isn’t necessarily, “Let’s put educational materials in GitHub”, but rather, “Let’s facilitate a culture of spontaneous-but-structured collaboration and improvement.”  

[Greg Wilson]
Create/post
Version control
Share with others
Reuse

Access/mark
Provide incremental feedback
Monitor progress of individual contributions to teams

Course materials

Assignments—team & individual

Read/access
Contribute (via pull requests, issues)

Incremental uploads
View other student/team activities

Key
Traditional learning management system
GitHub
Understanding Bots through a Model of Regulation (CSCW 2017)

Maryi Arciniegas-Mendez
maryia@uvic.ca
@maryiarciniegas
Self regulation bots

Felix bot

Felix helps you stay focused on the tasks that really matter to you today. Just tell him your top goals for today, and then cross them off your list as your day advances. It’s a small action, but a great productivity boost.

Felix only works for those who DM him. Not a team todo list, it helps you plan your day effectively.

Every weekday, Felix will remind you at the beginning of your day and when you are online to do a quick planning of your day.

Commands available:
- `start` to start your day.
- `add` to add more tasks to your day, such as `add Send Q1 financial report`.
- `show` to see today's work.
- `done` after completing a task, such as `done 1`.
- `done all` to mark all today's tasks as done.
- `settings` to update some of my config, such as the morning ping
- `feedback` to give Felix feedback, such as `feedback Send me weekly summaries`
Co-regulation

Nikabot

Every day Nikabot asks each member of your Slack team one question: “What project did you work on yesterday?” She then gathers the information and creates accurate Gantt charts and reports with the data split by time, project and person.

When you add Nikabot to your Slack team, she'll give you a link to your team's console. She will also deliver periodic reports to let all your team members know what everyone else is working on, increasing team-wide awareness.

Nikabot offers a hands-free, zero management approach.
Shared regulation bots

http://meekan.com/slack/
Learning from existing research

Concerns?
- **Fragmentation** of content and communication
- **Barriers** to community building
- Big data needs **“thick data”**

Opportunities?
- **Learning** is essential to productivity
- **Automation** and AI show great potential to improve knowledge work productivity
Software developers are the **knowledge workers** of tomorrow....

“Program or be programmed”
Douglas Rushkoff
Additional References (see slides for others)


Acknowledgements

Alexey Zagalsky,
Georgious Gougios,
Chris Parnin,
Fernando Figueria Filho,
Daniel German,
Maryi Arciniegas.