24th Annual Research Review

April 4-6 2017

Evaluation of multitasking and work interruptions in software projects

Alexey Tregubov

University of Southern California

Motivation

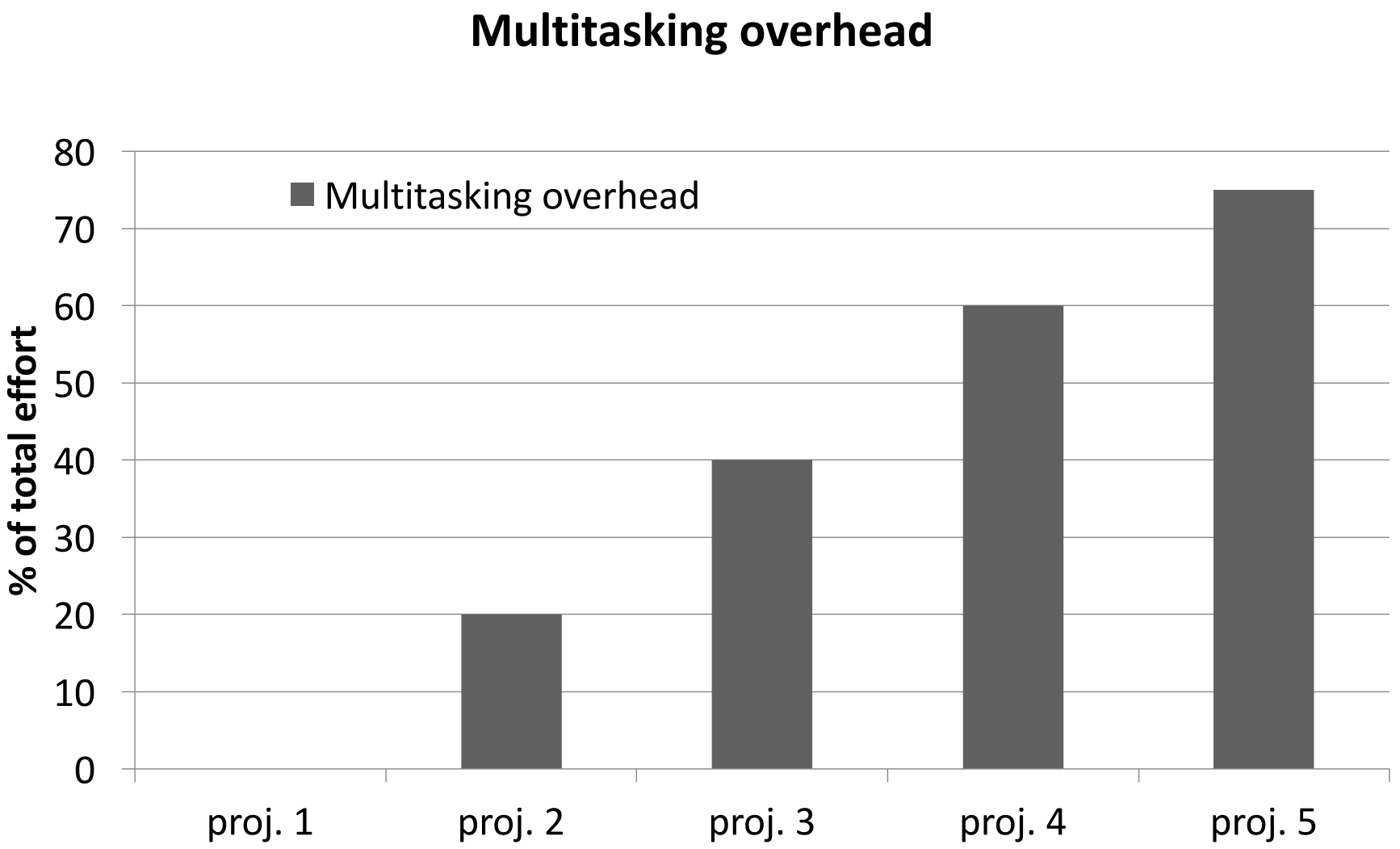
- Cross-project multitasking overhead is often observed in
  - Matrix organizations - resources are shared between several projects for better resource utilization.
  - Multiple releases of a product – if a product is released more than one time, resources are shared between maintenance of previous releases and a new version (typical for small mid-size teams).
  - System of System (SoS) environments – in SoSs, if a constituent system is developed for several customers (e.g. different software distributions/releases for each customer), resources are shared between different contexts. Context here is a customer-specific requirements, success-critical stakeholders, and everything that makes each system installation unique.
- Cross-project multitasking overhead is often not accounted when projects are estimated. This may cause underestimation in project planning.

Problem statement

In environments where software developers work on multiple projects, effort and schedule can be underestimated because of the following:

- Existing cost and schedule estimation models do not explicitly account for cross-project multitasking overhead.
- When work is planned and estimated by experts multitasking overhead is not taken into account.

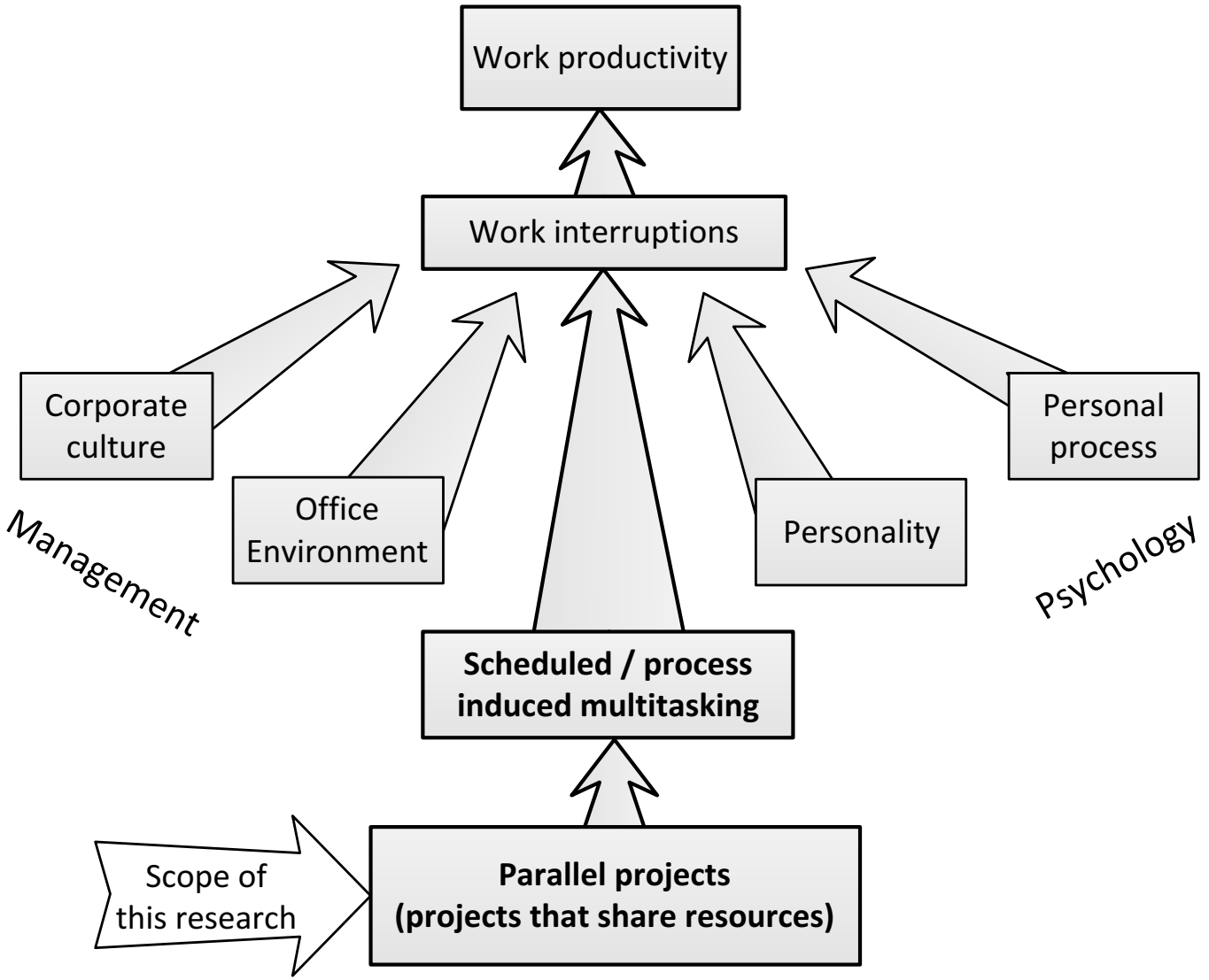
Gerald M. Weinberg's heuristic



Research questions and hypothesis

- Q1. Is the cross-project multitasking overhead (effort, number of interruptions) linearly correlated with the number of projects?
- Q2. What is the quantitative effect of cross-project multitasking on development effort?

Multitasking in work environment



Cross-project multitasking in software development

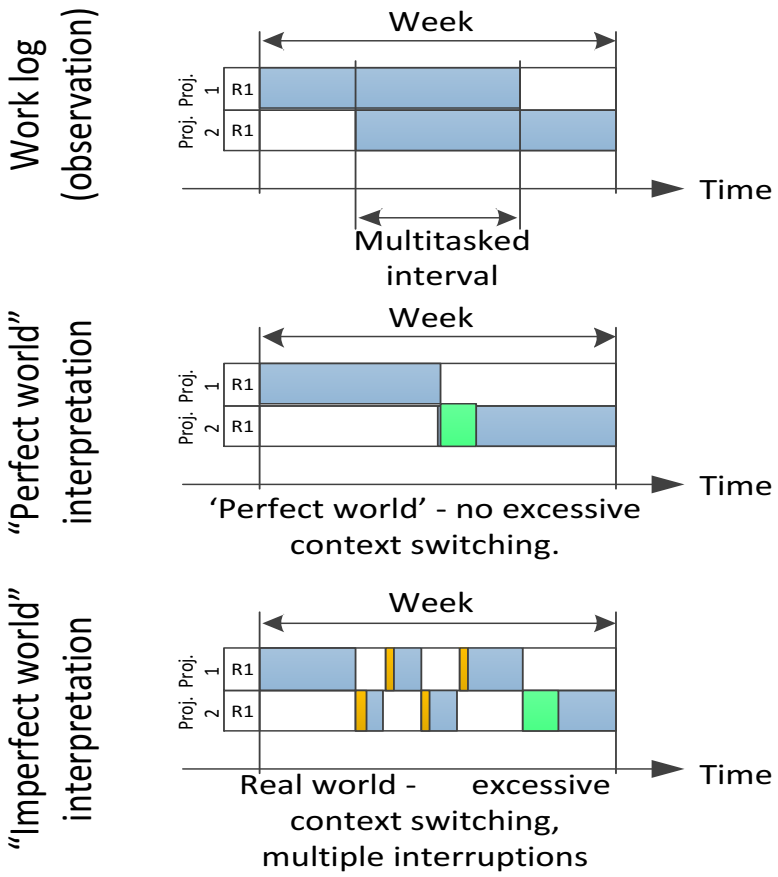
- When developers work on several projects at the same time (over a week or even a day), they switch between them spending some time on context switching.
- Not all context switching is bad, but we only focused on excessive switching between different projects, which is interrupting work and causing productivity decline.
- Context switching between projects cost/time consists of
  - **physical switching** – switch between repositories, DBs, servers, etc. takes time
  - **cognitive context switching** –getting into 'flow mode' takes time
- Multitasking overhead is not explicit in work log, so it needs to be evaluated. Work log analysis algorithm counts interruptions evaluates overhead in work logs.

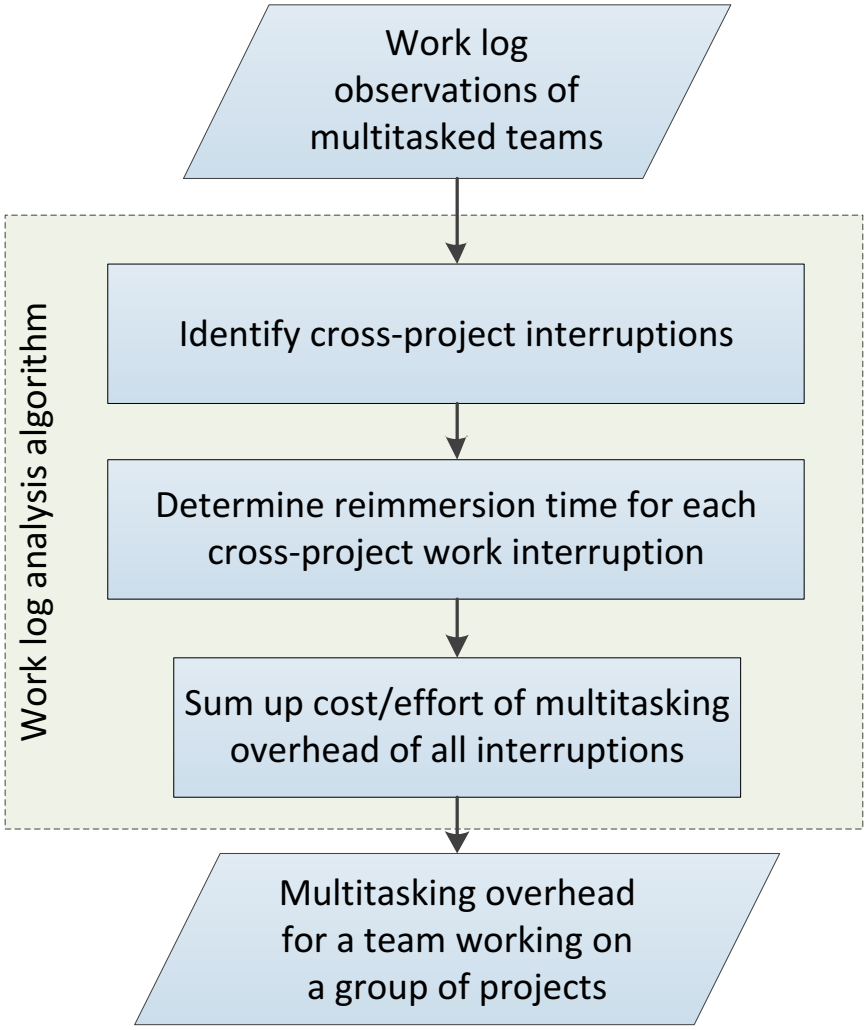
Research methodology

- A mixed methods approach (both qualitative and quantitative approach) has been selected for this research. The research is based on work log, schedule, and source code observations collected from two sources:
  - Student class projects (CSCI577 software engineering class)
  - Industry projects (from "MSS-Holding" Co.Ltd.)
- A work log analysis algorithm determines multitasking overhead for each instance of a work log

Work log analysis algorithm overview

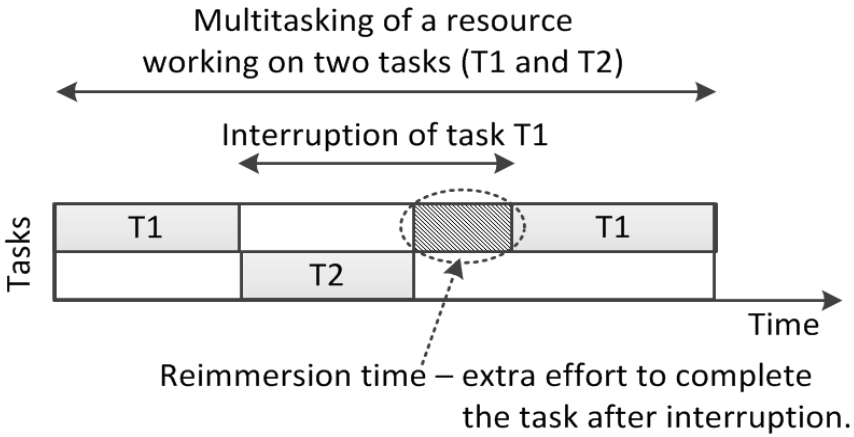
- Computes how often each task was interrupted over the week
- Computes reimmersion time of each interruption

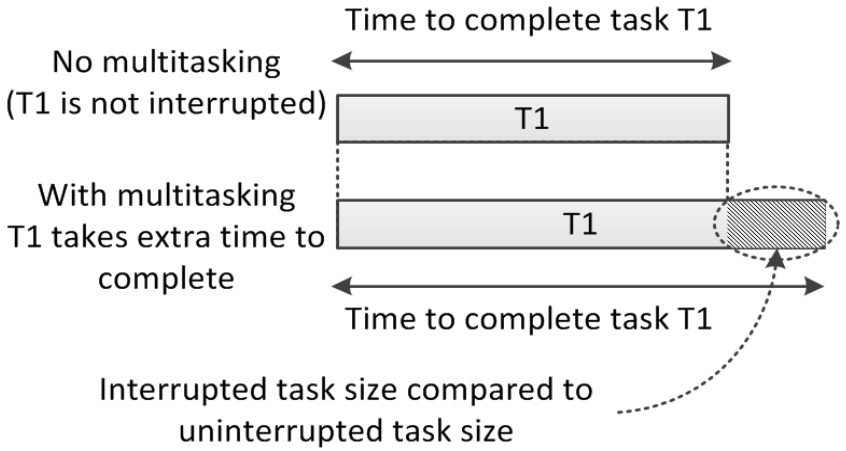




Interruptions and reimmersion time

- Reimmersion time determines the cost of work interruption



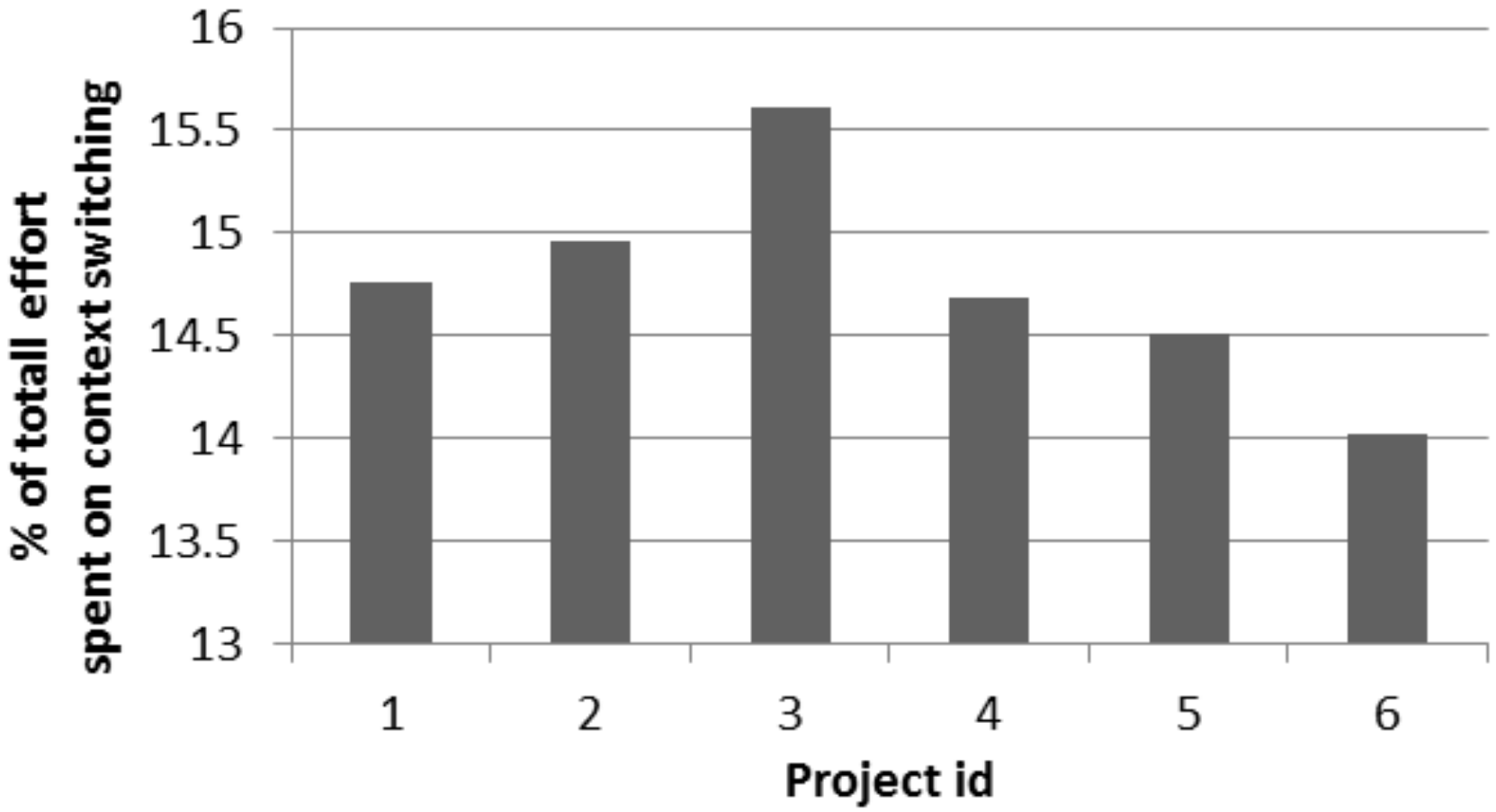


- Modeling the reimmersion time
  - Constant value (model parameter)
  - Self-evaluated values from subjects
  - Variable reimmersion time based on interruptions conditions
    - Task complexity
    - Length of interruption
    - Cause of interruption (self-inflicted vs. external)
- Reimmersion time range: 5 minutes – 2 hours

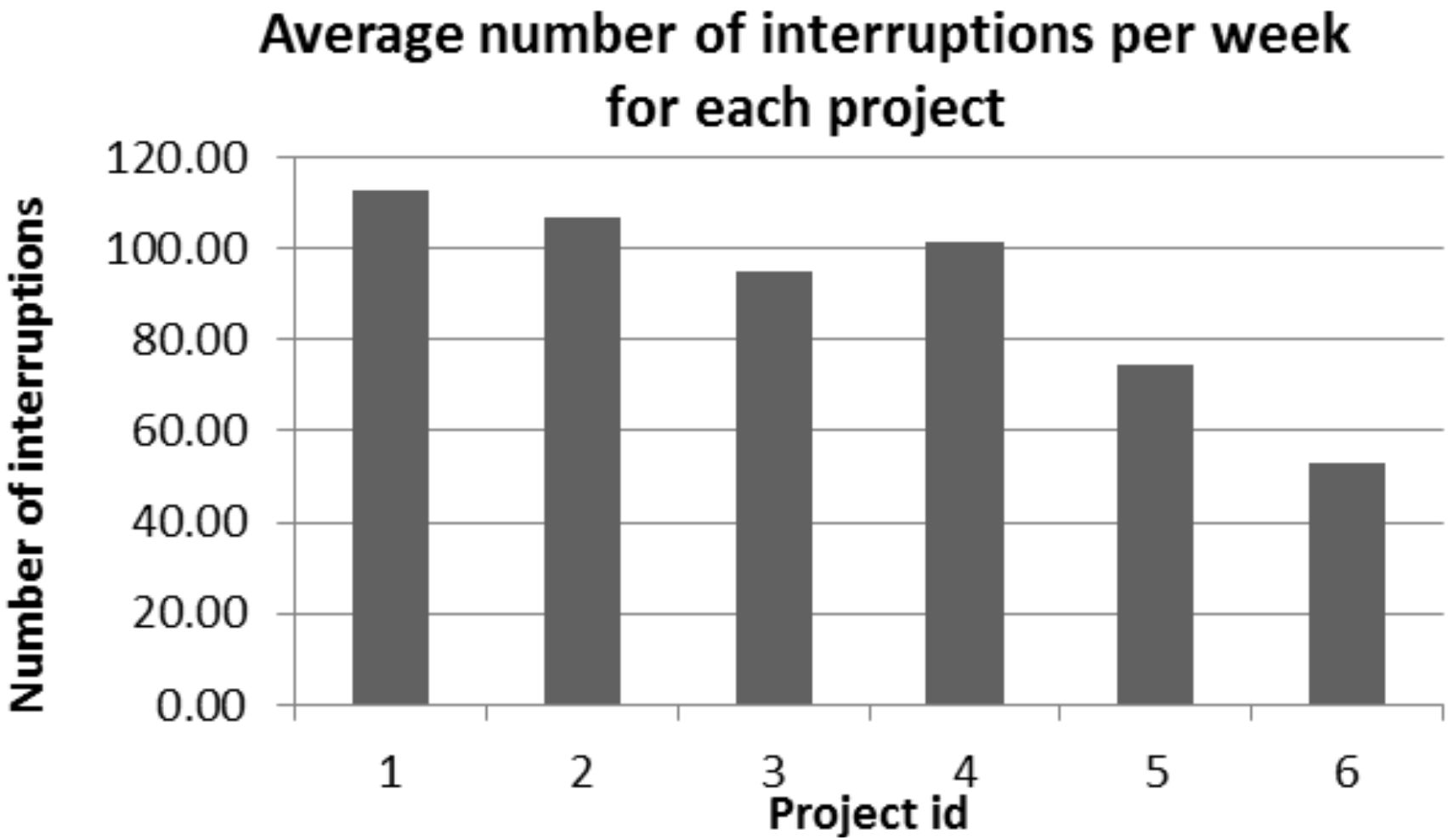
Data collection

- Industry projects:
  - 6 projects, 81 software developers
  - 1 year of work logs
  - Work logs in JIRA – hours reported daily for each task in each project, schedules were updated daily.

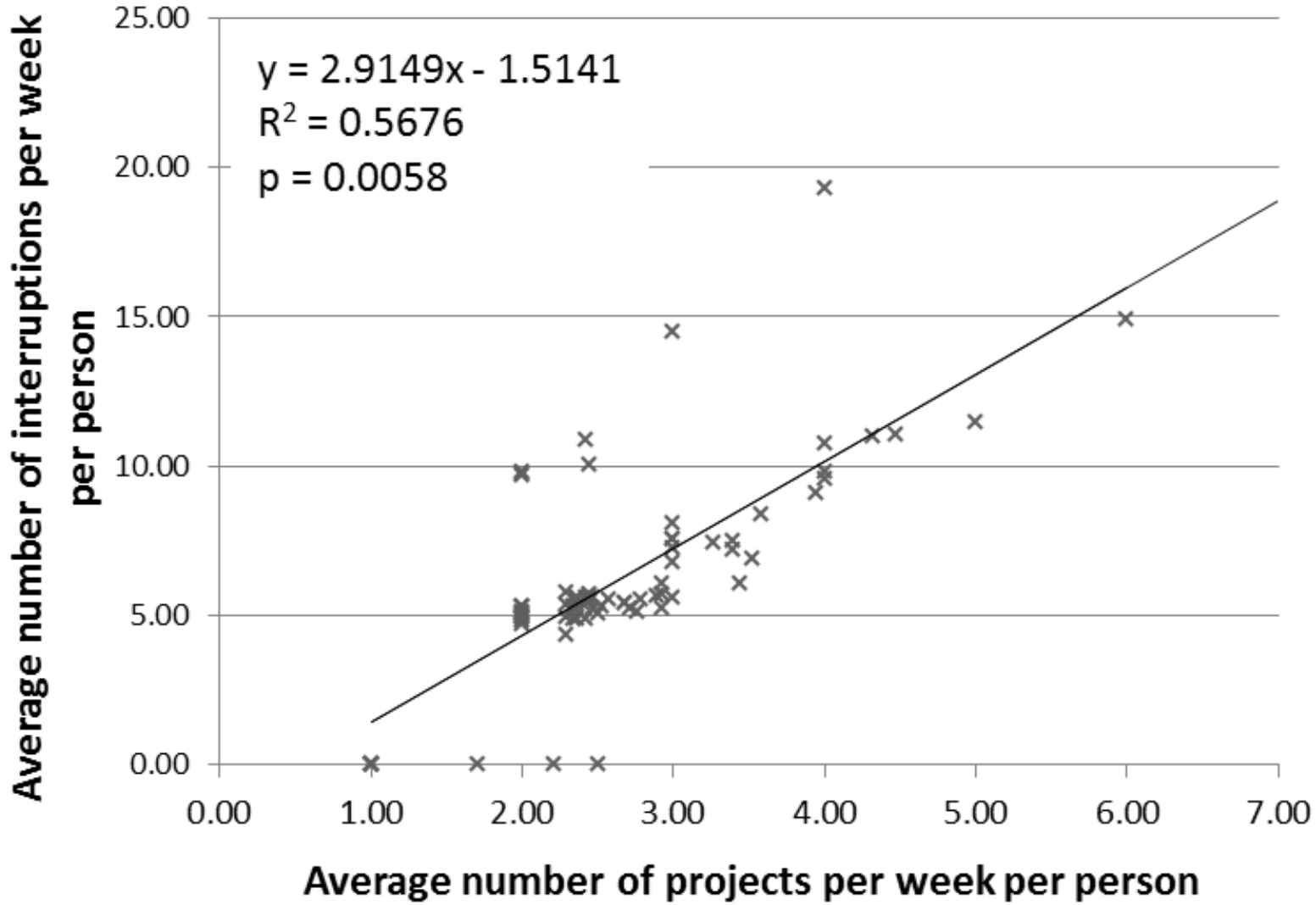
Results: multitasking overhead



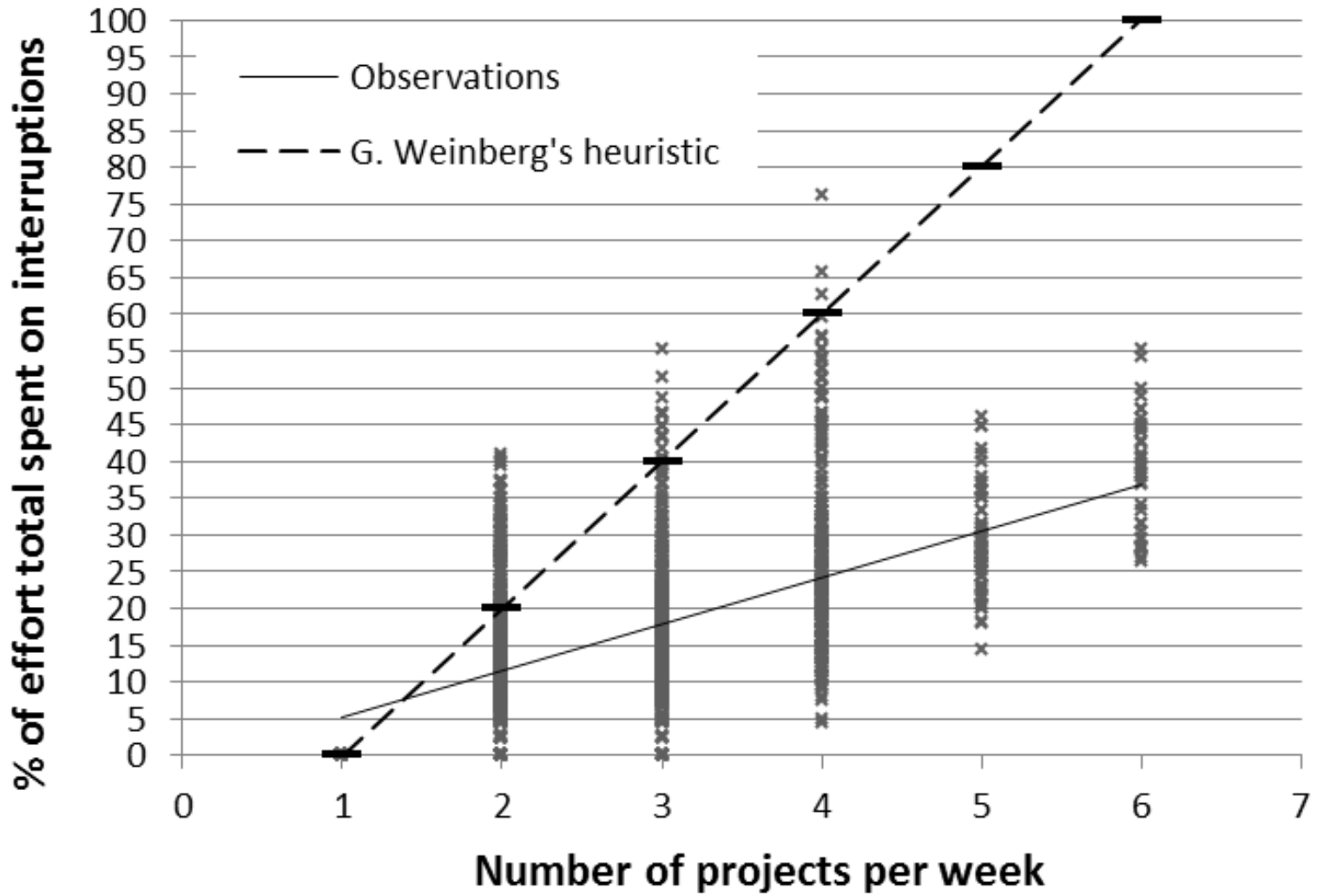
Results


Average number of interruptions per week for each project
Number of interruptions
120.00
100.00
80.00
60.00
40.00
20.00
0.00
1
2
3
4
5
6
Project id

Average number of interruptions



Results: observations vs. heuristic



Conclusion

- Analysed work logs of 81 software developers working on 6 projects for one year.
- Among all 6 projects at least 14% effort was spent on context switching between tasks from different projects.
- Developers who were involved in more projects tend to have more cross-project work interruptions.
- The linear correlation between the number of projects each resource is working on in one week and the number of interruptions exists but it is relatively weak.

Future work

- The impact of multitasking on effort can be integrated into parametric cost and schedule estimation models such as the Constructive Cost Estimation Model (COCOMO ®) for better effort estimation.
- The work log analysis tools, we used in the research, can be integrated with project tracking systems such as Atlassian Jira to provide real-time information about work interruptions and their impact on productivity.

Questions

Q&A

References

1. Meyer, D. E. & Kieras, D. E., 1997. A computational theory of executive cognitive processes and multiple-task performance: Part 2. Accounts of psychological refractory-period phenomena. Psychological Review, 104, 749-791.
2. Weinberg, G.M., 1993. Quality software management. New York.
3. DeMarco, T. and Lister, T., 2013. Peopleware: productive projects and teams. Addison-Wesley.
4. American Psychological Association, 2006. Multitasking: switching costs. URL: http://www. apa. org/research/action/multitask. aspx.
5. Dzubak, C.M., 2008. Multitasking: The good, the bad, and the unknown. The Journal of the Association for the Tutoring Profession, 1(2), pp.1-12.
6. Delbridge, K.A., 2000. Individual differences in multi-tasking ability: Exploring a nomological network. Unpublished Doctoral Dissertation, University of Michigan.
7. Tregubov, A. and Lane, J.A., 2015. Simulation of Kanban-based Scheduling for Systems of Systems: Initial Results. Procedia Computer Science, 44, pp.224-233.
8. Leonard-Barton, D. and Swap, W.C., 1999. When sparks fly: Igniting creativity in groups. Harvard Business Press.
9. Jett, Q.R. and George, J.M., 2003. Work interrupted: A closer look at the role of interruptions in organizational life. Academy of Management Review, 28(3), pp.494-507.
10. Tregubov, A. and Lane, J.A., 2015. Simulation of Kanban-based Scheduling for Systems of Systems: Initial Results. Procedia Computer Science, 44, pp.224-233.

References

11. Dingsøyr, T., Nerur, S., Balijepally, V. and Moe, N.B., 2012. A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software, 85(6), pp.1213-1221.
12. Boehm, B. and Turner, R., 2003. Balancing Agility and Discipline: A Guide for the Perplexed, Portable Documents. Addison-Wesley Professional.
13. Larman, C. and Vodde, B., 2008. Scaling lean & agile development: thinking and organizational tools for large-scale Scrum. Pearson Education.
14. Poppendieck, M. and Poppendieck, T., 2007. Implementing lean software development: from concept to cash. Pearson Education.
15. NDIA-National Defense Industrial Association, 2010. Top Systems Engineering Issues In US Defense Industry. Systems Engineering Division Task Group Report, http://www. ndia. org/Divisions/Divisions/SystemsEngineering/Documents/Studies/Top% 20SE% 20Issues, 202010.
16. Turner, R., Shull, F., Boehm, B., Carrigy, A., Clarke, L., Componation, P., Dagli, C., Lane, J.A., Layman, L., Miller, A. and O'Brien, S., 2009. Evaluation of Systems Engineering Methods, Processes and Tools on Department of Defense and Intelligence Community Programs-Phase 2 (No. SERC-2009-TR-002). SYSTEMS ENGINEERING RESEARCH CENTER HOBOKEN NJ.
17. http://www.businessdictionary.com/definition/matrix-organization.html
18. https://hbr.org/1978/05/problems-of-matrix-organizations
19. http://www.joelonsoftware.com/articles/fog0000000022.html
20. Mark, G., Gonzalez, V.M. and Harris, J., 2005, April. No task left behind?: examining the nature of fragmented work. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 321-330). ACM.