Team Portfolio Scrum: An Action Research on Multitasking in Multi-project Scrum Teams

Christoph J. Stettina[1,2(✉)] and Mark N.W. Smit[2]

[1] Centre for Innovation, Leiden University,
Schouwburgstraat 2, 2511 VA The Hague, The Netherlands
c.j.stettina@fgga.leidenuniv.nl
[2] Leiden Institute of Advanced Computer Science, Leiden University,
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

Abstract. Multi-project agile software development is a relatively new area of research. While original Scrum caters to co-located teams working on a single project, multi-project Scrum teams are a day-to-day reality, especially in small organizations. Multitasking across projects is frequently associated with loss of effectiveness, but this assumption is not sufficiently supported by empirical evidence. In order to better understand the phenomenon, we review existing literature across scientific domains and execute an action research project. Our findings show that the Team Portfolio Scrum (TPS) practice designed to support multitasking across projects is perceived to be useful, but with an associated increase in overhead.

Keywords: Agile software development · Scrum · IT project governance · Project portfolio management · Task-switching · Multitasking

1 Introduction

Should agile teams work on multiple projects simultaneously? While Scrum provides an example of how to execute individual software projects outside of plan-driven bureaucracies, the search for new organizational forms continues [1].

Scrum has been widely associated to cater well for a sweet-spot of co-located project teams working on a single project, with a pre-defined project scope and budget [2,3]. In particular, it stresses the need for teams to work on a single product per sprint [4]. Nevertheless, working on multiple projects during each sprint is a common reality [5]. For example, small companies with a small contract value and a large customer base are likely to accept multiple projects at the same time. Also, projects can be simply too small to fully occupy a team for the duration of a sprint. However, despite common sense across practitioners and anecdotal evidence implying a decrease in efficiency, there is little empirical evidence on teams working on multiple projects in parallel and the impact of multi-tasking.

H. Sharp and T. Hall (Eds.): XP 2016, LNBIP 251, pp. 79–91, 2016.
DOI: 10.1007/978-3-319-33515-5_7

In this article we (1) review empirical evidence on multitasking and task switching across different scientific domains, (2) propose the practice of Team Portfolio Scrum (TPS) and the role of the Team Portfolio Owner to help lowering costs of task switching, and (3) execute an action research project to understand the challenges of its introduction in practice.

2 Background and Related Work

While our understanding of Scrum in individual projects is quite elaborate, there is comparably little research on agile methods in multi-project and multi-team organizations [1,6]. In particular, the majority of literature on agile software development assumes an environment where a software developer or team works on only one project at a time. While most reports advise this [7], there is little empirical evidence on agile teams working on multiple projects. To illustrate our case, Table 1 shows an overview of the agile methods discriminated by either a single team or multiple teams that are working on either a single project or multiple projects. In this paper we address the case of *Team Portfolio Scrum (TPS)*, the case where a single team works on multiple projects simultaneously.

In order to better understand the challenges of multitasking across different projects we will now review existing evidence across the fields of (1) software development, (2) psychology, and (3) management science.

Table 1. Overview of agile methods across different organizational contexts

	A single project/product	Multiple projects/products
Single team working on	*Scrum*	*Team Portfolio Scrum*
Multiple teams working on	*Program Management & Scrum*	*Portfolio Management & Scrum*

2.1 Software Development: Interruptions and Multiple Projects

Existing software development literature generally considers task switching to be a wasteful practice that should be prevented whenever possible [8–10].

Working on multiple software projects: A common argument against multi-project development is that projects produce a revenue stream for the company at the time of their completion. Finishing them sequentially maximises revenue, because most often the revenue diminishes over time [11]. Another argument is that switching between tasks (e.g., across projects) is considered as waste [9]. Concrete numbers on the waste are hard to find; practitioners claim a small production increase from going from one to two projects (70 % to 80 % effectiveness) and a steady decline in effective hours when adding more projects: 60 % with three, 45 % with four and 35 % with five simultaneous projects [8]. A study amongst 64 high tech firms suggests two simultaneous projects is optimal [12].

Task and resource allocation practices: A challenge frequently mentioned by practitioners is that a team working on multiple projects is burdened with making decisions on which project to prioritise. Lehto and Rautiainen [13] describe governance challenges identified in a middle-sized software company. The role of product owner was described as too much for 5 co-located teams and was divided into three roles with split responsibilities: Solution managers (commercial), product owner (technical), resource owner (resource). Nocks [14] describes the practice to create very small sprints that match the amount of work per project, but this is countered by the large overhead of meetings per Scrum sprint. Other sources [15] discuss the need to let the different project managers negotiate the time allocation, and the need for one person to manage the final priorities of the projects the team works on and call this a Product Owner. If a team works on multiple projects, the team should work from one backlog during the sprint, containing work items from multiple projects [16–19].

Interruptions in software development: Van Solingen et al. [20] found that every time a software developer is interrupted by others (e.g., individuals from other or own project team), it costs on average fifteen minutes to get back to focus on the task he/she was performing. Parnin and DeLine [21] found that besides the initial delay, the quality of code produced following an interruption is lower, which corresponds to the residual impact found by psychology studies.

2.2 Psychology: Interruptions and Task Switching

Experimental results on switching between simple tasks: Research has found that task switching is not simply a cost in time going from one task set to the other [22]. Instead, the impact of a task switch consists of three components [23]: (1) the passive removal of the previous task context, (2) preparation for the new task, and (3) a residual impact. The removal of a task's context and preparation for a new task constitute the primary costs measured between the execution of different tasks. The residual impact is measured as an increased response time and sometimes increased error rate.

Experimental results show consistently that switching is more difficult if a complex task is involved. Results are ambiguous for the comparison of switching from a simple to a complex task and vice versa. Some experiments show that switching from a simple task to a complex one has increased primary costs [24]. Others show that switch costs are mostly determined by the task that is switched from [22]. This suggests that the residual impact is mostly determined by the previous task and that the results for the primary costs are ambiguous. Further, it matters what kind of stimuli and responses both tasks consist of. In the case where there is no overlap at all in stimuli and responses, task switching costs have found to be zero [25]. The more the stimuli and responses overlap, the greater the impact of the task switch. Furthermore, it has been found that when performing two task switches shortly after each other, switching back to the first task has higher switch costs than switching to a third, unrelated task [26].

Complex tasks in knowledge work: Compared to the daily tasks of knowledge workers, the tasks in the controlled experiments performed by psychology researchers are of a very simple nature, even the more complex ones. This is of course to make them controllable and repeatable. With these simple tasks, switch costs are measured in order of milliseconds. The kind of real life switches that knowledge workers perform are several orders of magnitude more complex, how does the psychology research map to this?

Some researchers suggest the results for contrived tasks can be generalized to more complex tasks [26]. We found that in all studies using a combination of simple and complexer tasks, the latter had higher associated switch costs; this was found already as early as 1927 [27]. One can assume that the switch cost increases based on some function over the complexity of the task, however we could not find such a function. Some practitioner sources claim effects in the order of minutes which is an indication of such a function and the relation between task complexity and switch cost.

Task-switching between similar tasks is known to increase stress [28]. Possible causes might be increasing the number of deadlines because of working on more tasks and decreasing the time available to meet the deadlines because of decreased productivity. On the other hand, [29] found that people very commonly self-interrupt, which might be a form of self-protection, decreasing fatigue and increasing performance [30]. In general, there exist various opinions on the effects of stress on performance [31].

Interruptions, work contexts and office spaces: Interruptions are omnipresent in the work of knowledge workers. Mark et al. [32] report that a knowledge worker spends on average only 12 min uninterrupted in a work context. Another study found knowledge workers spend very little time in one context and are interrupted before completion 57 % of the time after which they tasks are resumed on average after 25 min [33]. Interruptions are often harmful, to a large degree because it takes time to get back into a task or project [34]. Tasks, when interrupted, take longer and have increased error rates [35]. However, interruptions do not necessarily imply a context switch. Interruptions that lead to a switch between working spheres (e.g., two unrelated projects) are in general far more disruptive than interruptions from within the same sphere [32]. Further, Mark et al. [32] show that while co-located individuals (e.g., in open offices) face more interruptions in general, distributed individuals feel more free to engage in interruptions on topics that are actually unrelated to their work [32].

3 Research Objectives

While existing literature recommends minimizing the amount of concurrent projects, this might not be feasible for small companies depending on a large number of clients. Especially small companies are likely to pursue many projects simultaneously to keep their customer base satisfied.

Based on existing empirical evidence we may conclude that: (1) working on multiple external projects increases interruptions and work pressure as team members have to deal with requests from multiple Product Owners, (2) such interruptions by Product Owners are expected to be far more disruptive compared to the more frequent interruptions by team members working in the same project context, (3) the context switching penalty of task switching decreases performance and lowers code quality, and (4) penalty largely depends on similarity and complexity of tasks.

Considering these facts we can assume that distraction, uncertainty and context switching are likely to increase especially if priorities across different projects are not clear. Literature suggests that teams working on multiple projects should work from one backlog during the sprint [16–19]. However, such task prioritization practices have been reported to be difficult to establish [13–15].

As such we pose the following research question: *What barriers can be met and what benefits can arise from introducing a task prioritization practice to support a team working on multiple projects in parallel?*

4 Research Method and Conduct

In order to appropriately understand the dynamics of small organizations pursuing multiple projects in context, a complex social phenomenon, we decided to conduct an exploratory action research in the context of a real organization. Action research (AR) is designed to create knowledge by organizational change through a collaboration between researchers and practitioners [36]. It does so by diagnosing the current state, bringing about guided changes and reflecting on the results to create theory.

To ensure a credible research approach we applied the five principles of Canonical Action Research [37], as follows: *1. Researcher-client agreement:* The research has been executed as part of a 2 year collaboration with the university. This ensured the collaboration of the company and provided the possibility to bring about change as part of daily routines as well as very frequent observations. *2. Cyclical process model:* We adopted the five-stage process model of Diagnosing, Action Planning, Action Taking, Evaluating and Specifying Learning. One full cycle was completed. *3. Theory principle:* The theoretical ambition is to understand task prioritization and coordination practices in agile teams. *4. Change through action:* We supported the case company throughout the entire project. The second author was a full-time employee at the company facilitating workshops and discussions. *5. Learning through reflection:* Throughout the project meetings and workshops have been initiated to stimulate discussions among developers and management by the second author.

Case selection: The study was performed at a Dutch software company building bespoke custom software for customers. The company consisted of about 20 employees, half of which were software developers. The company had two teams working for two or more Product Owners nearly all the time. A known

challenge at the beginning of the research. The company had a flat structure, informal work environment and an open office space. Next to building custom websites for customers there are two in-house products. The team works on one of those products but a majority of the time is spent customizing the product implementation per customer. Greenfield development is rare. Maintenance is often urgent, with deadlines of one or two days being common. Work load was always high because of too few developers. The company has been endorsing Scrum from the beginning, however, struggled with its' implementation due to many parallel projects and customer requests.

Data collection and data sources: In order to build up an adequate understanding of the organization in context and throughout the action research project we used the following data sources:

Observations: While embedded within the company, the second author was able to observe the relevant practices at the company, including: (1) daily stand-ups, (2) portfolio meetings, (3) '*master stand-ups*', (4) planning sessions, and (5) development activities. We conducted structured observations on 41 occasions.

Semi-structured interviews: Next to informal discussions we conducted a total of 19 semi-structured interviews. Three types of interviews were executed: (1) Diagnosing and scoping, (2) mid-term, and (3) post-action interviews. The interviews were conducted with the management and development teams.

Questionnaires: We used bi-weekly questionnaires to create satisfaction graphs for involved practitioners over time (cf. [38]). The short questionnaires consisted of several questions using Likert-scales and open fields for additional remarks. Eight rounds of questionnaires have been collected with staff members.

Data analysis: In contrary to traditional passive qualitative research, the action itself provides a primary origin of interpretation [36]. To support the reflection among researchers and involved actors all observation notes, interviews and questionnaires were fully transcribed and used in discussions.

5 Action Research

5.1 Diagnosing

Diagnosing started in May 2013 and lasted until December. To understand the context, interviews were held with employees across all roles, resulting in a description of current roles and mapping of involved domains of practice. Generally the reported problems constituted a lack of structured process connecting the development to portfolio level decisions. On average the company has been working on two 'very small' projects and 30 to 40 'individual' to 'tiny' sized projects a year. When the company grew it became very difficult to keep an overview and coordinate these projects effectively. Further, the small teams were linked to multiple POs exposing them to discussions due to conflicting customer priorities.

During interviews and discussions with developers and management we identified a number of issues: *(1) Development staff is highly distracted, to a large degree caused by discussions with multiple POs. (2) Little connection of portfolio to strategy. Little coordination across the portfolio leading to suboptimal and unprofitable choices. Portfolio decisions are made by developers. (3) Poor knowledge management, resulting in overhead and posing a danger to project continuation. (4) Daily maintenance shifts help get maintenance tasks done while keeping most of the resources focused, however, overhead for working on unknown projects is high. (5) Hard to keep an overview and deliver work promised to clients.*

5.2 Action Planning

Following the diagnosis, management acknowledged that improvements were necessary. In discussions with all actors it became clear that a team having to deal with multiple POs does not work well because developers end up making decisions about portfolio priorities for a large part of their time. It was concluded that removing the portfolio decisions from the development staff and limiting interruptions of staff by management will likely reduce the distraction of team members and improve portfolio decisions. In early June 2014 an initial plan was developed based around the following proposals: (1) Introduce agile portfolio management, (2) introduce stable teams, (3) work with true Scrum sprints, limit task switching, and (4) improve company-wide knowledge management.

Based on that and hints found in practitioner's literature (cf. [39]), we designed the *Team Portfolio Scrum (TPS)* practice and the *Team Portfolio Owner (TPO)* role to support the implementation of portfolio management. TPS is based on a one week Scrum cycle including the usual Scrum practices (e.g., Sprint and review, daily stand-ups, retrospectives) in which the PO is replaced by the TPO. The practice follows characteristics of agile portfolio management [1] by (1) adding transparency of resources and work items through a Portfolio board, (2) close collaboration based on routines and artifacts enabling frequent feedback-loops across teams and management, (3) commitment to strategically managed portfolios, and (4) team orientation. In Table 2 we summarize the description and responsibilities of the new role.

5.3 Action Taking

Before introducing the TPO role, a portfolio team, a portfolio board and a team portfolio backlog were introduced. The TPO role was appointed from one of the three POs previously working with the team. On June 25 a workshop was held with the development team and management staff. Initial resistance arose among the developers as the goal was initially defined around increasing engineer productivity. In response the goal was rephrased to remove mid-sprint management interruptions and portfolio level responsibilities from the engineers.

Action taking began on June 30, 2015 with a reiteration of goals during a lunch presentation. The implementation of the practice was reflected throughout the project in Retrospective sessions staring on July 4. On August 15, 2014,

Table 2. The team portfolio owner role

Team Portfolio Owner:
Person responsible for the success a team's project portfolio. Analogously to a Product Owner, the Team Portfolio Owner (TPO) sets the priorities across projects during a Sprint. TPO shields team members from internal politics and different customers pushing project priorities
Responsibilities:
• Coordinates inter-project priorities with the team, portfolio team and customers
• Takes task switching penalties into account in discussions with the team and the Scrum Master
• Channels multiple projects into a single team backlog
• Guards inter-project priorities (1) when scope changes are needed (work taking longer/shorter or urgent other work, and (2) during the sprint planning meeting as the team negotiates work items
• Single channel of communication towards the team, lessening distractions (outwards communication is at the team's discretion)
• Attends the portfolio meetings to align priorities with company strategy
• Attends the daily stand-ups to keep up to date with the progress of the team

the team leader left the team for reasons unrelated to the project which had a big impact on the morale of the team as became visible in our satisfaction graphs. The action research continued with the practice being perceived as useful. The following time line outlines the execution of the project:

– **May 6, 2013:** Diagnosing: Scoping interviews
– **December 2, 2013:** Action Planning: Discussion of improvement initiatives
– **April 15, 2014**: Initial presentation of action plan to all employees
– **June 9, 2014:** Introduction of portfolio management (portfolio board)
– **June 24, 2014:** Preparation meeting with management and appointing TPO
– **June 25, 2014:** Workshop with development team and management
– **June 30, 2014:** Action Taking: Introduction of TPS practice
– **July 4, 2014:** Retrospective after first sprint
– **October 6, 2014:** End of Action Taking, beginning of evaluation
– **January 1, 2015:** Company wide implementation of TPS

5.4 Evaluating

The Team Portfolio Owner (TPO) role was adopted company-wide by our case company three months after the action research was completed. We consider this as an indicator for the success of the project. We now return to the research question in order to evaluate the action research.

Barriers to a team portfolio task prioritization practice: *Additional overhead:* In the case company, the hours of a manager, including the TPO, can not be billed to clients: *"As to whether it [the TPO role] is overhead, yes, per definition, because the work isn't billable to the client.* This is related to the way clients are billed at this company: actual booked development hours. Other methods exist that are much more suitable for Scrum [40]. However these methods assume one project per sprint. Billing might be a general problem for multi-project software development as is it hard to predict the proportions of the sprint for each customer in the face of scope changes.

High workload: The TPO reported his high workload at several occasions, especially towards the end of the action: *"..in the beginning I had much more time to do proper backlog management."*; *"it is extremely busy to fulfill this role."*

Benefits to a team portfolio task prioritization practice: *Better adherence to company strategy:* Due to the oversight the TPO can make better decisions in coordination across the entire portfolio. Yet, choosing the right projects can be difficult for a small company: *"For existing customers we basically have to do everything, we can't choose to not do a project. It is useful to decide on new customers though."*

Removing portfolio level decision making and conflicting decisions from multiple POs: This benefit of the introduced role functioned very well from the beginning, as confirmed by observations and multiple actors. Before introducing the practice, developers had to make decisions and were blamed for those. When asked about what to do when a task threatens achieving the sprint goals, a developer commented: *"I go directly to the TPO. The TPO manages what tasks get dropped. This works very well."*

Limiting interrupting requests from multiple POs: Before the change POs would often come to a developer's desk asking questions, planning work and lobbying for projects. As a developer comments: *"It is easier for developers to defend themselves.? and ?[the situation] improved. We have more breathing room because of the experiment. We can be more focused on software development."*

Specifying Learning. *Not more than one large context switch per day:* In our case organization the developers reported a benefit from the introduced TPS practice. However, also the number of parallel projects increased. Recommendations we found in literature deviate between two [12] parallel projects as an optimum, and not more than one large context switch per day - thus five projects per week. However, this largely depends on the homogeneity of the assignments. Here it is for the TPO and the team to discuss what a reasonable number of projects is according to: (1) familiarity with the project (architecture, code standards), (2) homogeneity (domain, application type), and (3) urgency.

TPO needs sufficient mandate: For fast resolving of issues, the TPO needs to have a complete mandate for choosing between the customers in the current sprint. A team member said at the beginning: *"The role itself has too much responsibility, at least too much for what the current TPO is mandated for.*

This adds a step between the management process: the team signals a problem to the TPO, the TPO needs to consult with the PO to make the decision. This means extra overhead for certain tasks. The TPO should define the priorities and shield the developers from the outside." The POs appreciated this delegation of responsibilities at later stages of the project: *"I liked it that the TPO could make the choices."*

Collaboration of TPO and POs: The introduction of the TPO role had a strong impact on the interaction of POs and teams. The POs had previously direct access to the teams, and had to go through the TPO as a *Master Product Owner* now. It took time to go through the TPO for planning or urgent maintenance: *"For me as a PO, the effect was that the planning was less fine grained, which was something I had to get used to since I'm a control freak. [About closing the scope] The smaller projects and maintenance are really hard to plan."*

TPO and the Portfolio Team: Knowing the inter-project priorities is very important for this role. The project portfolio board is the primary tool for the transfer of this information from the Portfolio Team to the TPO. Attending the Portfolio Management Meeting gives the TPO additional information and the possibility to discuss the priorities. The TPO said: *"The weekly portfolio management meeting is very important for this role."*

Limitations. There are two main limitations to this research: First, we present the results of a single action research study. Credibility of AR lies in knowledge generated and tested in practice [36]. Generalizations and external credibility from such AR studies depend on rich storytelling as well as application of AR guidelines such as CAR [37]. Second, with one developer leaving the team composition changed. This resource problem impacted the team, both in getting more work and lowering morale. However, many action research projects take an unpredicted course while still providing considerable scientific value [37].

6 Conclusions

In this paper we report on our experiences in introducing a task prioritization and coordination practice in Scrum teams executing multiple projects simultaneously. For teams operating in small companies such as the one presented here it is difficult to follow traditional Scrum as they are directly exposed to commercial pressure and customer needs. As such we address an under-researched scenario outside the *'agile sweet-spot'* [40] by linking Scrum to a portfolio management practice [1].

Despite the challenges encountered during this 17 months project, such as a team member leaving the team, the practice was perceived as useful by all participants and adopted company-wide after the project. The TPS practice helped our case organization to align tasks to strategy and limits interrupting requests to developers by appointing a dedicated Team Portfolio Owner.

To practitioners this paper provides the template of a concrete task prioritization practice, the barriers and benefits of its implementation. To academia, we

contribute to understanding of new and more agile organizational forms. We add a literature analysis describing the existing body of knowledge on interruptions and task-switching across the domains of software development, psychology and management science. As such we lay the groundwork for further investigations to quantify the effects of task prioritization and coordination practices in Scrum.

Multitasking seems unavoidable. The presented practice helped to run more projects simultaneously, however, the involved actors should be aware that it comes at a high cost. Companies need to make good strategic choices regarding resources and their allocation to stay viable and sustainable.

Acknowledgments. We thank all the participants for generously contributing to this study.

References

1. Stettina, C.J., Hörz, J.: Agile portfolio management: An empirical perspective on the practice in use. Int. J. Proj. Manage. **33**(1), 140–152 (2015)
2. Azizyan, G., Magarian, M.K., Kajko-Matsson, M.: Survey of agile tool usage and needs. In: Agile Conference (AGILE 2011), pp. 29–38. IEEE (2011)
3. Kruchten, P.: Contextualizing agile software development. J. Softw.: Evol. Process **25**(4), 351–361 (2013)
4. Deemer, P., Benefield, G., Larman, C., Vodde, B.: The scrum primer (2010). http://assets.scrumtraininginstitute.com/downloads/1/scrumprimer121.pdf. Accessed 12 Aug 2014
5. Payne, J.H.: Management of multiple simultaneous projects: a state-of-the-art review. Int. J. Proj. Manage. **13**(3), 163–168 (1995)
6. Marchenko, A., Abrahamsson, P.: Scrum in a multiproject environment: An ethnographically-inspired case study on the adoption challenges. In: Conference of Agile, AGILE 2008, pp. 15–26. IEEE (2008)
7. Highsmith, J.: Agile Project Management: Creating Innovative Products. Pearson Education, Boston (2009)
8. Wheelwright, S.C.: Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency, and Quality. Simon and Schuster, New York (1992)
9. Ikonen, M., Kettunen, P., Oza, N., Abrahamsson, P.: Exploring the sources of waste in kanban software development projects. In: 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2010), pp. 376–381. IEEE (2010)

10. Braun, E.: Lean/agile methods for web site development. Online-Weston Then Wilton **29**(5), 58 (2005)
11. Krebs, J.: Agile Portfolio Management. Microsoft Press, Richmond (2008)
12. McCollum, J.K., Sherman, J.D.: The effects of matrix organization size and number of project assignments on performance. IEEE Trans. Eng. Manage. **38**(1), 75–78 (1991)
13. Lehto, I., Rautiainen, K.: Software development governance challenges of a middle-sized company in agile transition. In: Proceedings of the 2009 ICSE Workshop on Software Development Governance, pp. 36–39. IEEE Computer Society (2009)
14. Nocks, J.: Multiple simultaneous projects with one extreme programming team. In: Agile Conference, 5 pp. IEEE (2006)
15. Wiseman, G.: Multiple projects, one agile team (2007). http://www.infoq.com/news/2007/12/multiple-projects-one-agile-team. Accessed 12 Aug 2014
16. Kathuria, M.: Happy marriage or divorce : What happens if single scrum team has to handle multiple projects (2013)
17. Levison, M.: Scrum on a small team with multiple "projects" (2013). https://groups.google.com/forum/#!topic/scrumalliance/8j-5V_Cl2aI. Accessed: 12 Aug 2014
18. Dinwiddie, G.: Combined backlog for multiple projects (2007). http://blog.gdinwiddie.com/2007/12/03/combined-backlog-for-multiple-projects/. Accessed 12 Aug 2014
19. Friedman, J.: Subprojects: Many projects, one team; one project, many teams (2013). http://blog.assembla.com/AssemblaBlog/tabid/12618/bid/98674/Space-Manager-Many-projects-one-team-One-project-many-teams.aspx. Accessed 16 Jan 2015
20. Van Solingen, R., Berghout, E., Van Latum, F.: Interrupts: just a minute never is. IEEE Softw. **15**(5), 97 (1998)
21. Parnin, C., DeLine, R.: Evaluating cues for resuming interrupted programming tasks. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 93–102. ACM (2010)
22. Wylie, G., Allport, A.: Task switching and the measurement of switch costs. Psychol. Res. **63**(3–4), 212–233 (2000)
23. Meiran, N., Chorev, Z., Sapir, A.: Component processes in task switching. Cogn. Psychol. **41**(3), 211–253 (2000)
24. Mayr, U., Kliegl, R.: Task-set switching and long-term memory retrieval. J. Exp. Psychol. Learn Mem. Cogn. **26**(5), 1124–1140 (2000)
25. Waszak, F., Hommel, B., Allport, A.: Task-switching and long-term priming: Role of episodic stimulus-task bindings in task-shift costs. Cogn. Psychol. **46**(4), 361–413 (2003)
26. Mayr, U., Keele, S.W.: Changing internal constraints on action: the role of backward inhibition. J. Exp. Psychol.: Gen. **129**(1), 4 (2000)
27. Jersild, A.T.: Mental set and shift. Arch. Psychol. **14**, 81 (1927)
28. Wetherell, M.A., Carter, K.: The multitasking framework: the effects of increasing workload on acute psychobiological stress reactivity. Stress Health **30**(2), 103–109 (2014)
29. Lenox, T., Pilarski, N., Leathers, L.: The effects of interruptions on remembering task information. Inf. Syst. Appl. Res. **5**(4), 11 (2012)
30. Keick, K.E.: Cosmos vs. chaos: Sense and nonsense in electronic contexts. Organ. Dyn. **14**(2), 51–64 (1985)

31. Diamond, D.M., Campbell, A.M., Park, C.R., Halonen, J., Zoladz, P.R.: The temporal dynamics model of emotional memory processing: a synthesis on the neurobiological basis of stress-induced amnesia, flashbulb and traumatic memories, and the yerkes-dodson law. Neural Plasticity **2007**, 1–33 (2007)
32. González, V.M., Mark, G.: Constant, constant, multi-tasking craziness: managing multiple working spheres. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 113–120. ACM (2004)
33. Mark, G., Gonzalez, V.M., Harris, J.: 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 (2005)
34. Czerwinski, M., Horvitz, E., Wilhite, S.: A diary study of task switching and interruptions. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 175–182. ACM (2004)
35. Eyrolle, H., Cellier, J.M.: The effects of interruptions in work activity: Field and laboratory results. Appl. Ergonomics **31**(5), 537–543 (2000)
36. Greenwood, D.J., Levin, M.: Introduction to Action Research: Social Research for Social Change. SAGE publications, Thousand Oaks (2006)
37. Davison, R., Martinsons, M.G., Kock, N.: Principles of canonical action research. Inf. Syst. J. **14**(1), 65–86 (2004)
38. Stettina, C.J., Heijstek, W., Fægri, T.E.: Documentation work in agile teams: The role of documentation formalism in achieving a sustainable practice. In: AGILE 2012, pp. 31–40. IEEE, Washington, DC (2012)
39. Singerman, D.: How does scrum work when you have multiple projects? (2009). http://stackoverflow.com/questions/412525/how-does-scrum-work-when-you-have-multiple-projects/413061\#413061. Accessed 12 Aug 2014
40. Hoda, R., Kruchten, P., Noble, J., Marshall, S.: Agility in context. In: Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications, OOPSLA 2010, pp. 74–88. ACM, NY, USA (2010)