Association for Information Systems
AIS Electronic Library (AISeL)

Research-in-Progress Papers | ECIS 2019 Proceedings

5-15-2019

FLOW EXPERIENCE IN SOFTWARE ENGINEERING: DEVELOPMENT AND EVALUATION OF DESIGN OPTIONS FOR ECLIPSE

Marc-Andre Kaufhold
University of Siegen, marc.kaufhold@uni-siegen.de

Christian Reuter
Technische Universität Darmstadt, reuter@peasec.tu-darmstadt.de

Thomas Ludwig
University of Siegen, thomas.ludwig@uni-siegen.de

FLOW EXPERIENCE IN SOFTWARE ENGINEERING: DEVELOPMENT AND EVALUATION OF DESIGN OPTIONS FOR ECLIPSE

Research in Progress

Kaufhold, Marc-André, University of Siegen, Siegen, Germany, marc.kaufhold@uni-siegen.de

Reuter, Christian, Technische Universität Darmstadt, Darmstadt, Germany, reuter@peasec.tu-darmstadt.de

Ludwig, Thomas, University of Siegen, Siegen, Germany, thomas.ludwig@uni-siegen.de

Abstract

In positive psychology, flow is described as a holistic mental condition in which an individual delves into an activity with full concentration. Even in software engineering, the promotion of flow experience fosters effects such as positive affect, improved learning, and higher product loyalty in computer-aided environments. However, from a practice-based perspective it is not obvious how to design ICT to support flow experience. With this paper, we, therefore, contribute concrete design implications, paving the way for a good flow experience in ICT. This paper begins by examining the current state of flow research in the field of Human-Computer Interaction. We then go on to present a study comprising the development and evaluation of design options that aim to support flow in integrated development environments such as Eclipse, one of the most prominent open-source IDEs. The findings reveal practical implications on the use of four flow design options for software engineering and are integrated into a preliminary research framework.

Keywords: Flow Theory, Human Computer Interaction, Design Options, Software Engineering.

1 Introduction

Flow experience describes a holistic mental condition in which an individual delves into an activity with a sense of active concentration, full involvement, and pleasure in the progression of the activity (Csikszentmihalyi, 1975). In the design-oriented and user-centered disciplines of Software Engineering (SE) and Human-Computer Interaction (HCI), the existing body of work analyzes the potentials of flow experience (Mahnke and Hess, 2014) and links the theory to the discourses around usability and user experience (UX). Previous research shows that flow in computer-aided environments contributes to positive affect (Hoffman and Novak, 1996; Chen, 2000), improved learning (Ghani, 1995; Guo *et al.*, 2013), and product loyalty (Mahnke, Benlian and Hess, 2015; Kaur *et al.*, 2016), among others. Thus, flow occasionally is described as the optimal user experience (Finneran and Zhang, 2005) and promises added value for the user and the organization offering flow-optimized ICT artifacts.

SE is particularly interesting for the examination of flow experiences: In a study about the individual perception of productivity, the participants mentioned "flow with few distractions and context changes" as a crucial criterion for a productive working day (Meyer *et al.*, 2014). The central tools of SE are usually integrated development environments (IDE) such as Eclipse. Considerations about work productivity should influence the design of IDEs (Murphy, 2014) but, especially considering beginners and semi-professional software engineers, also regard the adoption and continuance of use (Sun and Jeyaraj, 2013). Moreover, in the current state of flow research, a lack of connection between concrete design

options and theoretical components persists (Mahnke, Benlian and Hess, 2015) to directly address business-relevant performance indicators (e.g., the information system's ability to prevent users from switching to competitors). It, therefore, seems necessary to investigate design options to support flow experience in work environments – in this case using the field of SE with Eclipse – and factors contributing to the initiation and continuance of flow episodes. Hence, this paper addresses the following research questions: *RQ1) What are design options for supporting flow experience in SE with Eclipse? RQ2) What factors contribute to the initiation, continuance, and finishing of flow episodes in SE?*

This contribution starts with a classification of flow experience, introduces existing models of flow experience in IS and HCI, and creates references to the discourses around usability and UX (section 2). Based on theoretical findings and empirical results, the paper presents the development (section 3) and evaluation (section 4) of design options for supporting flow experience in IDEs, before discussing the results (section 5) as well as contribution and outlook of this study (section 6).

2 Theoretical Background and Related Work

Using the search scope framework of Brocke et al. (2015, fig. 1), our literature review can be described as follows: A sequential search process was conducted on bibliographic databases (ACM Digital Library, Google Scholar), aiming at a representative coverage of literature. Based on the initial results of our keyword search (using combinations of "flow", "flow theory" and "SE", "IS", "HCI", also in their written-out forms), we applied backward and forward searches based on identified publications that focused on flow experience in SE, IS and HCI contexts. Only a small subset of the identified 83 research publications are included in this research-in-progress paper.

Flow theory in IS and HCI. *Flow* and its intensity can be characterized by nine components (Csikszentmihalyi and Nakamura, 2009): (1) A balance between perceived challenges of the activity and skills of the individual, (2) clear operational objectives, (3) immediate operational feedback, (4) merging of action and awareness, (5) concentration on the current activity, (6) perceived control over the operation and the environment, (7) loss of reflective self-consciousness, (8) distortion of temporal experience, and (9) autotelic experience. The last of these describes the motivation of a person to fulfill an activity for its own sake. Flow requires a balance of the perceived challenges and skills, whereby the intensity of both variables must be above average (Moneta, 2012). In the context of IS and HCI, existing flow models essentially differentiate *antecedents/preconditions*, the *experience* itself and *consequences/effects* of flow experience (Finneran and Zhang, 2005). Schaffer (2014) describes flow experience as a continuous loop of a) the screening of the situational opportunities for action, b) the selection and implementation of one action and c) the evaluation of performance feedback, whereby the successful execution of those steps is tied up with particular flow conditions. Furthermore, Finneran and Zhang (2005) explain the importance of differentiating between the individual (P), artifact (A) and task (T) to allow for a clear understanding of what each contributes with.

Research opportunity. A central challenge concerns the determination of Mahnke et al. (2015) that so far little practical guidelines about the designing of flow experience in IS and HCI contexts have been derived. One interpretation is that the quantitative research, which is dominant in flow contexts, leads to higher levels of abstraction, whereby the connection between concrete design options and theoretical components was investigated insufficiently. That is a good starting point especially for design-oriented research (Finneran and Zhang, 2005). In their study, Mahnke et al. (2015) identified design options of flow experience in the context of online shopping. Accordingly, appropriate *search and filter functions* can support the persecution of a current aim and *recommendation systems* can encourage the development of new objectives. Regarding the organization of information, few hierarchical levels, consistent design and layout, and a clear structure of webpage elements seem to support the required goal orientation of flow experience. An appropriate information quality and quantity of edited and user-generated content to avoid information overload (Kaufhold *et al.*, 2018) contribute to that as well.

Application domain. SE seems a suitable research field for flow. From a practical perspective, it is common that IS students conduct SE activities in their theses or projects. These often have varying SE competencies and work in temporally limited projects or contracts. Thus, training efforts and SE

productivity during a project are important success factors. As IDEs usually represent a crucial tool in SE, it seems desirable to investigate which requirements of flow experience exist within IDEs to reduce training efforts and to increase SE productivity. Due to practical reasons, such as local access, we started our research with Eclipse developers. Eclipse is a Java-based IDE, whose core is a programming workspace with an extendable plug-in system allowing, e.g., the integration of external resources: bug trackers, build management tools, project management suites, and software revision management. From a theoretical perspective, research indicates that "software engineering research lacks theory and methodologies for addressing human aspects in software development" (Graziotin, Wang and Abrahamsson, 2015). In a study about the individual perception of productivity with 379 professional software engineers, 50% of the participants mentioned "flow with few distractions and context changes" as a criterion for a productive working day (Meyer *et al.*, 2014). Flow for IDEs suggests four interaction patterns: an adapted task context, provision of recommendations, appropriate dialogues, and summaries of context relevant issues (Murphy, 2014).

3 Development of Flow Design Options in Eclipse

We conducted a 90-minute explorative creative workshop with six participants of the field of IS, of which five were students and one was researcher (4x high and 2x medium Eclipse experience). The aims were a) inspections into work practices of the participants within the use of IDEs, b) the identification of barriers and problems within the use of IDEs, and c) the discussion of requirements and solutions for supporting the efficient and smooth use of IDEs. The workshop was structured in three phases each consisting of brainstorming, reflection, and discussion. The general usage and perceived use barriers of IDEs were discussed in a general manner including brainstorming, reflection, and combination of ideas (phase 1). In a next step they were embedded in the frame of flow theory including its introduction and the discussion of ideas in the scope of flow (phase 2). The collected ideas were then discussed as solution-oriented concepts including brainstorming, reflection and combination of requirements, and potential solutions (phase 3). The workshop was recorded and the participants' ideas were captured with presentation cards (phase 1 & 2). The recording was transcribed for analysis, and the results were summarized using open coding. They were also evaluated regarding literature findings and the flow components. Combining all results, we developed four mock-ups using Pencil (http://pencil.evolus.vn).

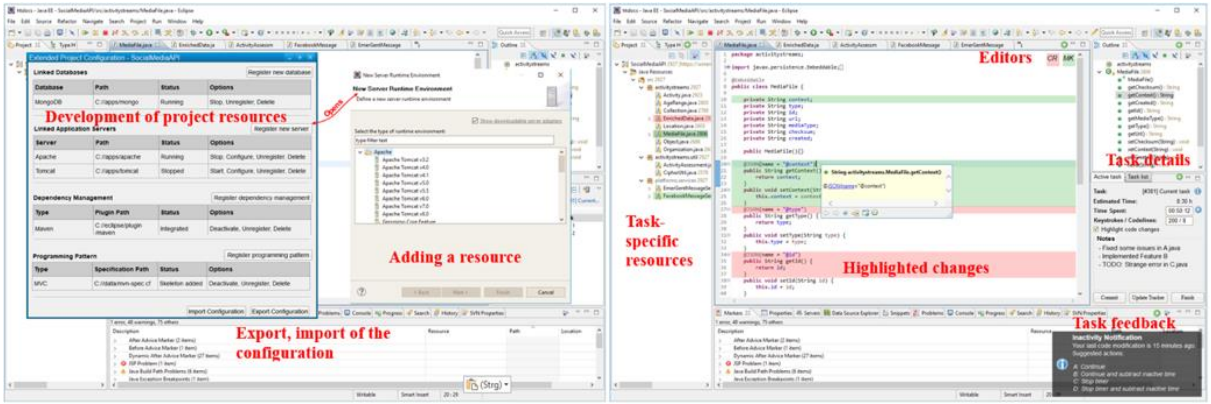

Figure 1. Flow Design Options (D1 left, D2 right)

Simplified resource management (D1). In the context of flow's *control* component, the participants expressed that the establishment and administration of a project and related technologies are associated with significant effort, e.g., when a new developer joins the project, or a new computer is configured for the work. To simplify management of necessary resources and to reduce barriers of the initiation of new flow episodes, the design option represents an interface, which allows the *setup of project resources*, such as libraries, databases, application servers, and programming patterns. Furthermore, the *export/import functions* indicate that resource configurations can be exported and imported in new environments as well as automatically configured. That reduces entry barriers, the potential for distraction (Schaffer, 2014) and supports the opportunity to get into a flow at the beginning of a task or project start.

Cross-application task context (D2). An essential issue discussed in the workshop was supporting the whole task processes by integrating a task repository or a bug tracker to avoid the effort and distraction caused by media disruptions and navigation steps. Eclipse Mylyn (Murphy, 2014) already integrates some requirements: The integration of external task repositories, hence the transfer of tasks and aims into the IDE and the reduction to *task-relevant resources* in the project explorer. Besides, a note functionality and time recording for the current task were desired. Furthermore, the participants of the workshop discussed aspects concerning collaboration, which affect the flow experience of an individual or a group as a collective: The awareness about handling and synchronizing joint files. Thus, the design option extends the Mylyn task view by *task details* such as time recording, which is supported by an absence *notification*, and a *memo function* for notes to support the usage of notices for change reports. Additionally, it illustrates *editors* and *highlights code changes* to encourage the discussion about awareness and collaboration functions. Regarding indirect collaboration, a reference to changes of precious revisions is integrated into the context menu of the editor. Thus, a cross-application task context without unnecessary media disruptions can be conducive to the maintenance of flow.

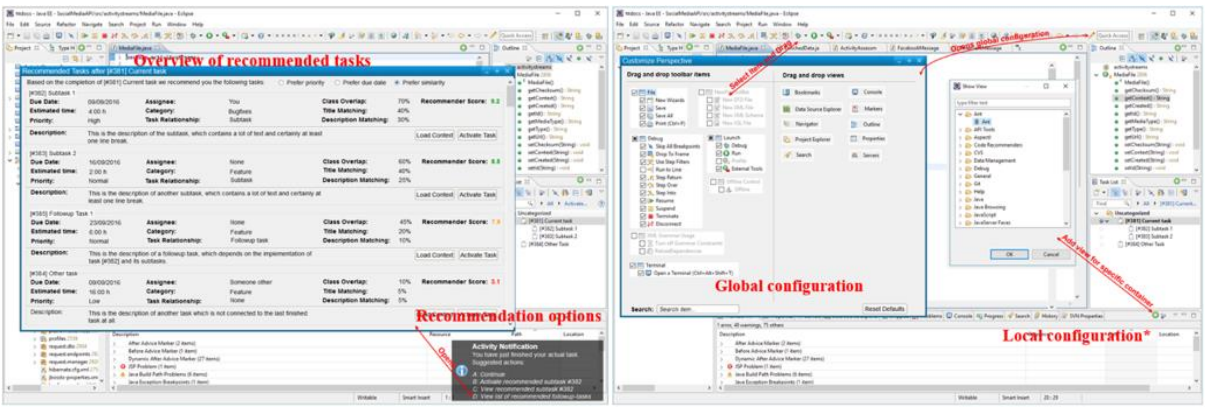

Figure 2. Flow Design Options (D3 left, D4 right)

Recommendation of following tasks (D3). In the context of flow's *feedback* component, the participants discussed the usage of a recommendation system, which suggests the user further similar tasks to be accomplished, based on the currently active or completed task. This is interesting in the respect that on the one hand, frequent context changes reduce the productivity and on the other hand, the generation of new aims, as in terms of new tasks, can contribute to structuring the working day (Meyer *et al.*, 2014). For flow in IDEs, Murphy (2014) also discusses the use of recommendation systems to a) provide a selection of accurate alternatives and b) strive for the automation of operations. Here, a challenge exists within the definition of which criteria, and in which (algorithmic) weighting of these criteria the actual recommendation for pursuing tasks can be derived. The aim of the design option is to support the maintenance of existing flow episodes and the initiation of new ones. After finishing or selecting a specific task, the design provides *recommendation options* of similar or relevant tasks via notification, which also allows opening an *overview of recommended tasks* as a list.

Simple functional configuration (D4). During the workshop, it was mentioned that Eclipse is overloaded with functionality. Most of the functional elements such as perspectives, views (tabs) and toolbars in Eclipse are therefore configurable but in an inconsistent manner. While a button for the addition of a perspective is available in the perspective management, views and toolbars must be configured in the menu bar. It seems appropriate to implement latter configurations also via buttons directly in the view and toolbar areas (local configuration; plus icons) to avoid any distraction from the actual task (Meyer *et al.*, 2014). Moreover, tools such as Mozilla Firefox supply a specific customization mode, whereby the available control elements can be moved to (or deleted from) the desired position on the browser. Allowing the configuration of the toolbar or even the whole perspective, considering toolbar items and views, via a button (global configuration; pencil icon) could be an added value for Eclipse regarding the consistency of operation (Mahnke, Benlian and Hess, 2015). This design realizes the two options of a) consistent *local configuration* of control elements at their respective position and b) *global configuration* of control elements to support a consistent operation management and therefore flow.

4 Evaluation of Flow and Design Options

The evaluation of our concepts comprised two steps: The conduction of a *survey* to evaluate the design concepts with users, developers, and usability experts in general, and *interviews* with software developers to assess the design options in more detail. For the survey, we designed a questionnaire containing both qualitative and quantitative questions and statements to query written evaluations of users, usability experts, and developers from the workshop. On the cover of the questionnaire, flow theory, Eclipse, and the goal of the study were introduced. Further, data concerning the age group, gender, and technological expertise of the participants was collected. The following four pages of the questionnaire followed the same pattern: First, a mock-up illustrated the design concept and the concept was explained in short. Then, two open-ended questions followed per concept. These questions concerned a) a general evaluation of the concept and b) a special aspect of it. Additionally, to further evaluate the concepts, there was a different number of statements per concept (cf. tables 1-4), which were related to the original flow components. The participants had to demonstrate to which degree they agree with the respective statement on a five-point Likert-scale. We had 21 questionnaire participants (F1 to F21) of different age groups (20-30 [10], 30-40 [5], 40-50 [1]), gender (male [12], female [3], not specified [1]), and technical expertise (user [1], usability expert [11], developer [15]). The questionnaires were used as guidelines for the in-depth interviews with developers to obtain more meaningful results with users from the target group. The basics of flow theory and the results of the explorative workshop were introduced to the interviewees. We had four interviewees (I1 to I4) with high Eclipse expertise. To document the results, the interviews were recorded, transcribed for further analysis, and summarized using open coding.

Resource management: countering efforts, frustration and time delay (D1). The results indicate that the proposed design option can promote flow by reducing efforts and frustration, but also reveal challenges regarding a concrete implementation (Table 1). The participants said it supports flow in situations such as the introduction of a new developer into a bigger project (I1). *Export and import functions* have an advantage over traditional instructions by saving recurring structures of the general project context and making them reusable (I2; I1). They also said that *management features* are useful in the context of, for example, managing server runtimes (I3) and that the main overview is simple in the current state and thus should not be complicated by too many additional options. On an affective level, the concept could also prevent distraction and frustration through an excessively long configuration time (I3). Although it potentially helps to install the components on the computer more easily (I4), the fourth participant assumed such an integration to be accompanied by license law issues (I4) and a high complexity when transferred to other platforms with different hardware and software configuration details (I4). However, I2 argued that default paths could be used here (I2) and that a semi-automatic process could enable specific configuration parameters to be adapted before the import process is performed.

An extended resource configuration supports administration during a project.	4,00
An export/import function facilitates the migration of resource configurations.	3,89
An extended resource configuration facilitates flowing operation of a new project.	3,74

Table 1. Average results of the questionnaire (D1)

Task context: supporting immersion due to reduced media disruptions (D2). The design option potentially facilitates flow by supporting immersion, e.g., due to reduced media disruptions (Table 2). However, not all proposed features were well received. It was emphasized that *time recording* supports the user's time consciousness on the one hand (I3) and the project manager who must manage the project resources on the other hand (I1). Regarding privacy, I4 was convinced that measuring time is a controversy accepted by some users without further objections and simultaneously considered to be disastrous by others. In contrast, another participant thought it appropriate to record time step by step for the respective activities (I2). The notification of recognized absence and options allowing a correction of already recorded working time enable a more precise time recording (I1) and further are a useful feature to react appropriately to the absence in the IDE (I4). Furthermore, the *memo function* was discussed in the context of time recording, which was perceived as an option for documenting progress in a summarized form (I1). Participant I4 argued that a user possibly overlooks the memo function and that therefore

it would make sense to highlight it in the form of a summary, modifications made, after an activity has ended. That would enable the user to efficiently generate an activity report for the bug tracker and the versioning software (I4). In total, the combination of memo function and time recording could help to maintain flow because the features are available at the same time and are located centrally at the user interface (I3). The interviewees highlighted the functions for strengthening *awareness for the own or the others' code activity*. These could enable an overview of activities (I4) while equally supporting the distributed programming (I1). However, users should be able to *uncheck* code modifications so that they are no longer highlighted (I1). Overall, it should be possible to filter by both editors and status, for example: to highlight all changes, to hide marked changes, to not highlight any changes, and to show the status before the changes. Regarding collaboration, it would also be useful if one could comment on the changes in the code. That would enable an editor to give explanations to other editors directly in the context so that it is no longer necessary to leave a note in the bug tracker (I1).

A memo function reduces the post-processing effort of documentation.	4.00
Emphasizing my changes of code supports me navigating my tasks.	3.86
Emphasizing foreign changes of code and cursors supports team collaboration.	3.71
The notification of absence supports me recording time correctly.	3.62
A process timer reduces the post processing effort of a task.	2.86
Quantitative attributes (keystrokes, code lines) motivate me working on a task.	2.14

Table 2. Average results of the questionnaire (D2)

Task recommendation: individual and situational demands (D3). After completing a task, the recommendation of following tasks potentially promotes the initiation of new flow episodes and reduces context switches but underlies individual and situational demands (Table 3). The interviewees saw the concept of *recommending tasks* as a suitable mechanism either to maintain the recent flow episode after an activity has ended or to initiate a new flow episode (I1; I3; I2). Especially in situations in which a multitude of tasks is available, the value of such a recommendation system is high (I3), and the decision-making autonomy of the user or supervisor remains despite the recommendations (I3). Furthermore, one participant emphasized that the value of the recommendation system depends on the scope of the project: In small projects, the developer could manually examine the tasks of his own area of expertise without any problems; however, in bigger projects and with a narrow specialization of the developer, the system is more likely to be useful (I4). Regarding the *relevance of the criteria* influencing the calculation of a recommended score, the due date represented the most important criterion, followed by both priority and similarity of the task (I3). The due date should especially be favored in the context of hard deadlines (I4). In soft deadlines, the relative usefulness of the similarity of a task to generate flow episodes increases. *User behavior and preferences* which influence the decision besides the actual recommendation should be considered in the recommender score (I1). Interviewees stated that a user may evaluate afterward whether he was in flow or liked a task and that similar tasks should be proposed. Also, the technical expertise may influence the selection process (I1). Such a system could be adopted not only for the user's self-selection, but also for delegation by supervisors to provide task queues for developers (I3).

The recommendation of subsequent tasks supports the workflow.	4.00
The due date of the task is suitable to weight subsequent tasks.	3.95
The priority of the task is suitable to weight subsequent tasks.	3.86
The recommendation of subsequent tasks supports generation of new goals.	3.55
A notification is suitable to recommend further tasks.	3.62
The similarity of a task is suitable to weight subsequent tasks.	3.29

Table 3. Average results of the questionnaire (D3)

Function configuration: eliminating distractions (D4). As the results indicate, the proposed design potentially helps reducing distractions and interruptions in the workflow of users (Table 4). According to the interviewees, the design options of local and global *function configuration* are useful supplements

to the status quo of Eclipse. Concerning local configuration, they said that it is intuitive to load desired views per plus-icon into tabs. And regarding global configuration, they anticipated that every developer is able to master the operating per drag and drop (I1; I4; I3). They further indicated that the concept reduces distraction and may support flow experience (I2) if the user interface must be adapted due to task challenges or operation errors (I3). One participant proposed the functionality of saving the different views collectively with the resource configuration (D1) in a *configuration file*. This configuration file could be imported to readjust the interface accordingly (I2). Such a configuration could be transmitted to friends, employees or other devices (I4). Furthermore, short summaries could describe the functions of the IDE in more detail (I2). In addition to the concept itself, the participants discussed the *usability* of Eclipse. They said that distributed configuration options were not implemented as expected, also regarding flow; an improved implementation remained desirable (I4).

Local configuration facilitates the adaptation of the development environment.	3.89
Global configuration increases the consistency of the operation.	3.89
Global configuration facilitates the adaptation of the development environment.	3.72
Local configuration increases the consistency of the operation.	3.68
Distributed configuration is appropriate to adapt the development environment.	2.89

Table 4. Average results of the questionnaire (D4)

Critical stances towards flow experience. One of the participants pointed out explicitly that the immersion character of flow experience represents a risk in specific professions or activities, for instance, in safety-critical domains (Reuter, 2018). Furthermore, it must be remarked that additional positive experiences can be the target of a good design: "Basically, I think that a sound mixture is best. Also, it depends on the specific task. If there is deadline pressure and my tasks must be completed, obviously, flow is best. However, as far as the daily life is concerned and if there is no time pressure. I prefer a mixture of arousal, control and relaxation. You cannot only experience flow but also need to do tasks that are not stressful. Still, I must admit that I feel best after coding if I really was in flow" (01:24:53, I1). Another suggestion was to instead examine the prevention and paths of negative experiences such as anxiety, concern, indifference and boredom (I1). It was pointed out that methodical disturbances, for example in the contexts of eye tracking and thinking aloud, must be eliminated to measure flow experience correctly to generate convincing and valid findings from the theory.

5 Conclusion and Outlook

Previous research proved positive effects of flow experience in IS environments (Chen, 2000; Guo *et al.*, 2013) and established a series of different models to operationalize flow experience (Finneran and Zhang, 2005). This paper examined design options to facilitate flow in the domain of software engineering, using the example of Eclipse, and therefore adopted a similar course to Mahnke et al. (2015) who discuss design options for flow experiences during online shopping. Combining the findings of literature and an explorative workshop, we developed and evaluated the design options of resource management (D1), task context (D2), task recommendation (D3), and function configuration (D4).

Key findings. Looking at software development with Eclipse from a practical perspective (RQ1) showed that efforts to configure functions (D4) and resources (D1) before and during a task can lead to distractions or interruptions of flow episodes. Meaning that more consistent function and resource configuration may reduce this problem. We also found that the use of a tailored task context (D2) – which is realized in Eclipse Mylyn and whose productivity advantages had already been analyzed scientifically (Murphy, 2014) – may be supported with a compact representation of task details, a memo function, and time recording. Here it became clear, analogous to (Meyer *et al.*, 2014), that time recording could be staggered to activities more precisely, but also could transmit the feeling of surveillance. It also became apparent that because of the individual nature of flow experience and situational factors the design must provide diverse opportunities for action. Thus, filter and search functions as well as personalization options in terms of a recommender system for subsequent tasks (D3) are required to make the design suitable for practical application. Furthermore, opportunities for action may be provided with context-

sensitive notifications. However, with their immediate-style interruptions (McFarlane, 2002) and without configurability they may challenge the user's attention and therefore disturb the flow experience.

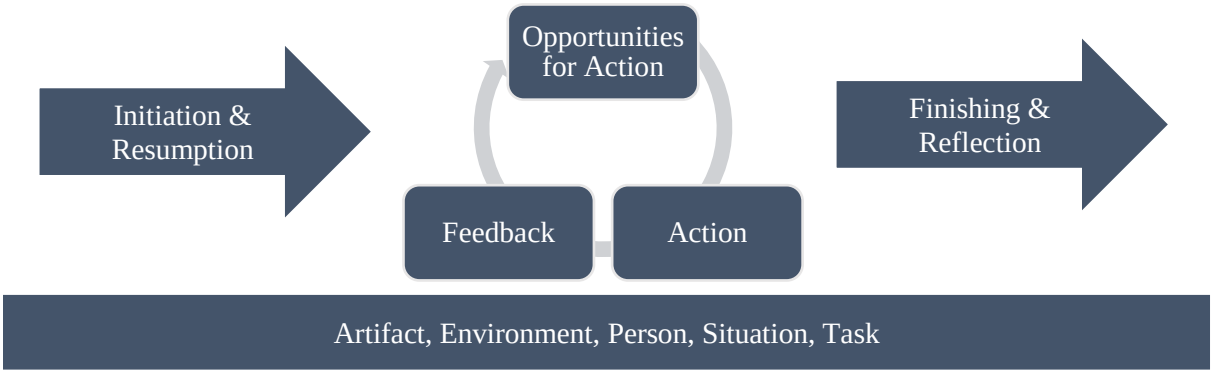

Figure 3. Preliminary research framework: embedding of the flow loop model based on Schaffer (2014) *in the steps of initiation & resumption as well as finishing & reflection*

We integrated our findings into a preliminary framework (Figure 3). We found that IT *artifacts* (along with *environmental* influences, *personal* preferences, *situational* demands and *task* characteristics) might not only contribute to the continuance, but also the initiation of new and resumption of interrupted flow episodes (RQ2). For the initiation of new episodes, task recommendation (D3) was evaluated. Generally, this means using different criteria to compute recommender scores for suitable follow-up tasks, based on the current or a selected assignment. Interruptions of flow episodes are important scenarios that should be analyzed systematically for two reasons. Firstly, because the attention energy for continuance of flow episodes is limited, as development tasks can be very complex and long-ranging. And secondly, because emergent, high-priority development tasks can influence the personal and resource planning. In the context of the cross-application task support (D2), the marking of code changes, the memo function and time recording could contribute to re-familiarize with the task context after interruptions. The discussed functions also affect the finishing of a task. While flow is characterized as a reflection-free episode (Csikszentmihalyi and Nakamura, 2010), after the finishing of a task, the developer is required to write a change report for the project management software where reflection and recapitulation of performed changes is required. With the assistance of reflection processes or the reduction of effort and frustration through insufficient technical support, the finishing of flow episodes could contribute to the growth of self-awareness and promotion of expertise (Csikszentmihalyi, 1990).

Limitations and future research. The presented design options are first drafts to evaluate the potentials of facilitating flow experience in the context of the integrated development environment Eclipse. In order to achieve more generalizable results, the study needs to examine additional IDEs in the future. Also, as a further limitation, the sample was rather small, partially due to the questionnaire's length and special audience of semi-professional developers. In future, the study should be enhanced by including professional developers and further stakeholders involved in the software development process. Furthermore, it is not only required to investigate the anticipation of flow experience, but also to verify whether the expected effects occur in practice (Tuunanen and Govindji, 2015). Therefore, as next steps, a proof-of-concept and a field study must follow to evaluate the quality of the proposed design artifacts. In doing so, the design options must (iteratively) reach the required degree of maturity and its implementation must be specified and conducted to validate the effects. In the long term, the examination of further design options (across fields) offers the potential to systematically compare domain-specific results and to develop requirements with a higher abstraction. This could contribute to closing the gap between purely theoretical flow constructs and case-specific design options. Since flow research already indicated an effect of flow on online learning continuance (Guo *et al.*, 2013), it seems promising to examine the impact of flow on learning during software engineering.

Acknowledgements. The research group *KontiKat* (Reuter *et al.*, 2017) was funded by the German Federal Ministry of Education and Research (BMBF) (no. 13N14351). The research centre *CRISP* was funded by BMBF and the Hessen State Ministry for Higher Education, Research and the Arts (HMWK).

References

Brocke, J. *et al.* (2015) 'Standing on the shoulders of giants: Challenges and recommendations of literature search in information systems research', *Communications of the Association for Information Systems*, 37(1), pp. 205–224.

Chen, H. (2000) *Exploring Web Users' On-line Optimal Flow Experiences*. Syracuse University.

Csikszentmihalyi, M. (1975) *Beyond Boredom and Anxiety*. San Francisco: Jossey-Bass.

Csikszentmihalyi, M. (1990) *Flow: The Psychology of Optimal Experience*. New York: Harper & Row.

Csikszentmihalyi, M. and Nakamura, J. (2009) 'Flow Theory and Research', in Snyder, C. R. and Lopez, S. J. (eds) *Oxford handbook of Positive Psychology*. Oxford University Press, pp. 195–206.

Csikszentmihalyi, M. and Nakamura, J. (2010) 'Effortless Attention in Everyday Life: A Systematic Phenomenology', in Bruya, B. (ed.) *Effortless Attention*. Cambridge: MIT Press, pp. 179–190.

Finneran, C. M. and Zhang, P. (2005) 'Flow in computer-mediated environments: promises and challenges', *Communications of the Association for Information Systems*, 15, pp. 82–101.

Ghani, J. A. (1995) 'Flow in Human Computer Interactions: Test of a Model', in *Human Factors in Information Systems: Emerging Theoretical Bases*. Ablex Publishing Corp., pp. 291–311.

Graziotin, D., Wang, X. and Abrahamsson, P. (2015) 'Do feelings matter? On the correlation of affects and the self-assessed productivity in software engineering', *Journal of Software: Evolution and Process*, 27(7), pp. 467–487. doi: 10.1002/smr.1673.

Guo, Z. *et al.* (2013) 'Promoting online learners' continuance intention: An integrated flow framework', *Information and Management*. Elsevier B.V., 53(2), pp. 279–295.

Hoffman, D. L. and Novak, T. P. (1996) 'Marketing in Hypermedia Computer-Mediated Environments: Conceptual Foundations', *Journal of Marketing*, 60(3), pp. 50–68.

Kaufhold, M.-A. *et al.* (2018) 'Mitigating Information Overload in Social Media during Conflicts and Crises: Design and Evaluation of a Cross-Platform Alerting System', *Behaviour & Information Technology (BIT)*.

Kaur, P. *et al.* (2016) 'Flow in context: Development and validation of the flow experience instrument for social networking', *Computers in Human Behavior*, 59, pp. 358–367.

Mahnke, R., Benlian, A. and Hess, T. (2015) 'A Grounded Theory of Online Shopping Flow', *International Journal of Electronic Commerce*, 19(3), pp. 54–89.

Mahnke, R. and Hess, T. (2014) *Flow Design Method : Four Steps towards Optimal User Experience*.

McFarlane, D. (2002) 'Comparison of Four Primary Methods for Coordinating the Interruption of People in Human-Computer Interaction', *Human-Computer Interaction*, 17(1), pp. 63–139.

Meyer, A. N. *et al.* (2014) 'Software developers' perceptions of productivity', *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014*, 2(Section 4), pp. 19–29.

Moneta, G. B. (2012) 'On the Measurement and Conceptualization of Flow', in *Advances in Flow Research*. Heidelberg: Springer-Verlag, pp. 23–50.

Murphy, G. C. (2014) 'Getting to Flow in Software Development', in *Systems, Programming, Languages and Applications: Software for Humanity (SPLASH)*, pp. 269–281.

Reuter, C. *et al.* (2017) 'Digitalisierung und Zivile Sicherheit: Zivilgesellschaftliche und betriebliche Kontinuität in Katastrophenlagen (KontiKat)', in Hoch, G. et al. (eds) *Sicherheit (DIAGONAL Jahrgang 38)*. Göttingen: Vandenhoeck & Ruprecht, pp. 207–224.

Reuter, C. (2018) *Sicherheitskritische Mensch-Computer-Interaktion: Interaktive Technologien und Soziale Medien im Krisen- und Sicherheitsmanagement*. Wiesbaden: Springer Vieweg.

Schaffer, O. (2014) *Crafting Fun User Experiences: A Method to Facilitate Flow*.

Sun, Y. and Jeyaraj, A. (2013) 'Information technology adoption and continuance: A longitudinal study of individuals' behavioral intentions', *Information and Management*. Elsevier B.V., 50(7), pp. 457–465.

Tuunanen, T. and Govindji, H. (2015) 'Understanding flow experience from users' requirements', *Behaviour & Information Technology*, 3001(November), pp. 1–17.