DO DEVELOPERS MAKE UNBIASED DECISIONS? – THE EFFECT OF MINDFULNESS AND NOT-INVENTED-HERE BIAS ON THE ADOPTION OF SOFTWARE COMPONENTS

Complete Research

Stefi, Anisa, Ludwig-Maximilians-Universität München, Munich, Germany, stefi@bwl.lmu.de

Abstract

Software reuse can lower costs and increase the quality of software development. Despite a large body of research focused on technical and organisational factors, there is still limited research on the software developers' perspective regarding software component reuse. Therefore, this paper investigates the developers' adoption intention to use existing software components. Information systems adoption research has extensively focused on the technological aspects and less on the individual factors. However, studying these individual differences is important, as research has shown that individuals do not always behave according to rational assumptions. This study analyses the adoption of software components based on the unified theory of acceptance and use of technology and extends the research model by integrating the not-invented-here bias and the concept of mindfulness to account for individual differences. An empirical study with 142 software developers was conducted to empirically validate the research model. The results show that performance expectancy, social influence and not-invented here bias play an important role in the developers' decision to adopt software components. Furthermore, findings show that a mindfulness state has a negative influence on the not-invented-here bias and it directly affects the intention to adopt existing software components.

Keywords: Unified theory of acceptance and use of technology (UTAUT), not-invented-here bias, mindfulness, software reuse, software components.

1 Introduction

The development of complex information systems requires large investments, making the success of software projects important for every organisation. Although information system development has become more mature in the last decade, software projects are often not on schedule, exceed costs, or are delivered with less functionality than initially requested. In fact, the Standish Group reports that in 2012, 43% of the projects were challenged, which means that they were not on schedule, over budget, or with less functionality, and that 18% of the projects were cancelled prior to the completion (The Standish Group, 2013). One strategy to improve software productivity is the reuse of software artifacts (Boehm, 1999). Software reuse is "the use of previously developed software resources from all phases of the software life cycle in new applications by various users such as programmers and systems analysts" (Kim and Stohr, 1998 p.113). As such, organizations do not have to "reinvent the wheel" and can resort to proven software components that have been developed internally or externally by a third-party (Ravichandran and Rothenberger, 2003). The benefits of using available software components in the software development process have led to an increasing availability of such ready-to-use components, thus playing an important role in the IT industry (Ayala et al., 2011).

Initial research on software reuse has focused on the technological issues (e.g. programming language support, creating and retrieving reusable artifacts, repositories, etc.), and only later non-technical factors (e.g. organization, processes, business drivers) were found to be important for the success of a reuse strategy (Ravichandran and Rothenberger, 2003). However, little research has focused on the factors influencing the adoption of software components from the developers' perspective (Mellarkod et al., 2007; Sojer and Henkel, 2010). Developers need to invest time and resources to estimate the benefits and drawbacks of using existing software components. Software developers that do not consider this decision face opportunity cost; the missed opportunity to profit from the quality and the time invested in implementing the existing code functionality. Such opportunity cost might also have an effect on the overall organisational performance, as the decision making of individuals influences efficiency and outcome at the project level. To facilitate the reuse of existing code, companies usually invest in a code repository within an organization, which requires upfront investments. Thus, understanding the adoption intention of such a technology is crucial for the success of the organization (Mellarkod et al., 2007).

This research analyses software reuse adoption from the individual developer's perspective. Developers are the users of the existing software components, which is why this work is grounded in the Unified Theory of Acceptance and Use of Technology (UTAUT) (Venkatesh et al., 2003; Venkatesh et al., 2012). This theory provides a holistic view on the most influential factors affecting the user's adoption of information technology (IT), and is grounded on previous models and theories. Usually, UTAUT is applied to organisational and consumer context, where technological expertise is not required. However, unlike in the traditional context, software developers have high technological expertise and can also implement the desired functionality themselves. As a result, software developers might build certain expectations about the software component, and might reject the adoption process, even when it could be beneficial. Hong et al. (2014) argue about the significance of the contextualized research in theory development. Thus, whereas UTAUT constructs have focused on the impact of the technology and social pressure, this research model is extended by adopting cognitive and psychological constructs relevant to the context of the study. Other researchers have also argued about incorporating psychological constructs and cognitive theories within the adoption context (Goswami et al., 2009). One of the psychological factors discussed within software reuse literature is the not-invented-here bias (NIH) which is also referred to as the tendency to "reinvent the wheel". By incorporating the NIH bias, the model accounts for developers' tendency to rely on self-developed components, rather than existing ones developed by third-parties. Additionally, the cognitive factor of mindfulness, defining a state of alertness in which decisions are based on the person's current needs, is included (Langer, 1989). Whereas cognitive styles have not directly shown to influence adoption and use of technology (McElroy et al., 2007), the cognitive state of mindfulness has been shown to lower the uncertainty related to the technology and to influence the information systems (IS) adoption decision (Sun and Fang, 2010).

Most research analysing the adoption and use of new technologies is based on the assumption of economic-rationalistic models and often lack the subjective perceptions of individuals (Fichman, 2004). Consequently, decision outcomes were attributed to general technology characteristics rather than the individual differences. Therefore, this study additionally to the UTAUT constructs, considers the cognitive processes and psychological factors that are not related to the technology per se. Thus, the intention of software developers to use existing software components and how a mindfulness state and the NIH bias contribute to this decision, are investigated. More specifically, this study answers the following research questions:

1. What factors influence developers to adopt existing software components?
2. How do a mindfulness state and the not-invented-here bias influence the adoption decision of software developers?

In order to answer these research questions, a survey with 142 developers was administered. The results contribute to the research literature, as they enrich the understanding of decision-making in general and especially within the software development context. They provide a new perspective on software reuse, which has practical implications for the overall performance of IS development.

The reminder of the paper is structured as follows. First, the theoretical background on software reuse, technology acceptance research, and the cognitive concept of mindfulness are described. Second, the hypotheses are derived and the research model is presented. In the subsequent section, the data collection procedure is discussed in detail, followed by the data analysis. Finally, the paper concludes by discussing the findings, as well as theoretical and practical implications.

2 Theoretical Background

This section will first briefly summarize the different views that have been taken to analyze software reuse in the literature. Subsequently, relevant details of the UTAUT and mindful theory will be presented. The section will conclude by pointing out the research gaps that this study addresses.

2.1 Software Reuse

Despite the importance of IS, software projects are frequently over budget, not completed on time or not delivered with the intended functionality (Sauer et al., 2007). Adopted from other engineering disciplines, reuse is seen as an important strategy in order to increase software development productivity and improve quality (Biggerstaff and Richter, 1987; Schmidt and Buschmann, 2003). Due to the reuse of software artifacts in many applications, the software components have been frequently tested and improved, resulting in software products that have fewer mistakes and are easier to maintain (Frakes and Kang, 2005). Moreover, adopting software reuse creates competitive advantages for companies, as software systems account for a considerable amount of costs (Frakes and Isoda, 1994). One major impediment to successful systematic reuse is often seen in the conflict created between the strategic organization's long-term objectives and the immediate objectives of individual project teams, such as meeting customer requirements on time and on budget (Fichman and Kemerer, 2001). Research has recognized that software reuse requires changes in the work of developers and organizations, and therefore some organizations are more likely to resist such an adoption (Sherif and Vinze, 2003). One of the inhibitors of software reusability recognized in the literature is the NIH bias, also known as NIH syndrome (Biggerstaff and Richter, 1987; Chapman and van der Merwe, 2008; Fichman and Kemerer, 2001; Griss, 1993; Sherif and Vinze, 2003). However, Heinemann et al. (2011) and (Haefliger et al., 2008) have found that within open source, reuse is achievable at high rates (more than 50%) and code is reused without modification known as black-box reuse. Whereas research has focused on the organisation's perspective, there has been little research from the individual developer's perspective. One of such studies based on TAM found out that the perceived usefulness and the developer's mindset about reuse were important determinants of developers' intention (Mellarkod et al., 2007). The software developer's intention to reuse software code was also investigated within the open source community based on the Theory of Planned Behaviour (TPB) (Sojer and Henkel, 2010). Pardue and Landry (2009) also analyse the individual intention, but focus on the influence of trust and social norms on the decision to reuse software components.

2.2 Technology Acceptance Research

One of the most prominent research streams of IS, is that of technology acceptance research. This research stream aims at explaining the users' adoption and usage of a new technology. To address this question, different theoretical models with origins in psychology, IS and sociology were adopted (Venkatesh et al., 2003). One of the core aspects within this research is the understanding of actual behaviour through intention. Thus, initially stated in the Theory of Reasoned Action (TRA) (Ajzen

and Fishbein, 1980) and elaborated further in the TPB (Ajzen, 1991), the user's actual behaviour is the result of the user's intention for the given action, attitude toward the action, his subjective norms as well as, argued by the TPB, the perceived behavioural control (Ajzen, 1991). Based on TPB and a stronger focus on technology, Davis (1989) proposed the Technology of Acceptance Model (TAM), which posits that the users' intention of using a new technology is influenced by the perceived usefulness and the perceived ease of use of the technology. Due to the different models adopted in the literature, Venkatesh et al. (2003) developed UTAUT on the basis of the literature review founded on seven theoretical models. According to UTAUT, *Performance Expectancy*, *Effort Expectancy*, *Social Influence* and *Facilitating Conditions* are the main factors influencing the intention to adopt a technology. These constructs are further moderated by *Age*, *Gender*, *Experience*, and *Voluntariness*. To address the consumer context, Venkatesh et al. (2012) developed UTAUT 2 and extended UTAUT with three new constructs that are: *Price Value*, *Habit* and *Hedonic Motivation*. This study analyses software component reuse from the perspective of individual developers and through the technology acceptance theoretical lens. Whereas UTAUT model has not been applied in this context, developers as potential adopters of the technology have been also considered in other studies within the IS adoption research (Mellarkod et al., 2007; Polančič et al., 2011; Sojer and Henkel, 2010; Woon and Kankanhalli, 2007).

2.3 Mindfulness

Mindfulness, at its roots, is a psychological concept that reflects upon the cognitive qualities of the individual (Langer, 1989). The key qualities of a mindful state of being involve: (a) openness to novelty; (b) alertness to distinction; (c) sensitivity to different contexts; (d) implicit, if not explicit, awareness of multiple perspectives; and (e) orientation in the present (Sternberg, 2000 p.12). Dane (2011) define mindfulness as "state of consciousness", rather than a quality that some individual possesses or lack. Mindfulness is distinguished by other similar related phenomena (e.g. absorption or flow), as it characterized by focusing the attention to the present moment, while considering a wide range of attentional stimuli (Dane, 2011). The concept of mindfulness has also been adapted to the organisational context (Weick and Sutcliffe, 2001). Butler and Gray (2006) argue that both individual and collective mindfulness, help to better design, manage and use complex IT systems reliably and efficiently. Prior research has considered mindfulness in the adoption of IT innovation (Swanson and Ramiller, 2004), within agile software development (Vidgen and Wang, 2009) or the case of managerial mindfulness in the adoption a new technology (Goswami et al., 2009).

In the technology adoption research, Sun and Fang (2010) have found out that mindfulness reduces uncertainty and influences the adoption behaviour. The mindfulness state was also investigated in the post adoption and continued use (Sun, 2011). Decision makers' mindfulness has been argued to diminish the bandwagon behaviour within technology adoption (Fiol and O'Connor, 2003). Similarly, mindful employees are less prone to the negative effect of information overload within business processes outcomes (Wolf et al., 2011). The state of mindfulness results in individuals that align their knowledge to their present contextual environment while considering alternative solutions. Mindlessness is the alternative state of "reduced attention resulting from premature commitment to beliefs that may not accurately reflect the phenomena at hand" (Butler and Gray, 2006 p.215). Thus, a mindless state drives individuals to perform routine actions, without questioning the current context or efficiency, making it difficult for individuals to react to the changing environment. Within software development, it is often the case that new technologies emerge and developers need to adapt to new technological and business processes. Therefore, it is relevant to consider the state of mindfulness in the adoption decision.

Prior studies analysing the intention to reuse software assets (including code snippets or design artifacts) are based on TAM (Mellarkod et al., 2007) and TPB (Sojer and Henkel, 2010). The UTAUT model is not yet adopted in such a context, and it is argued to provide more explanatory power (Venkatesh et al., 2003). Most interestingly, the relation between mindfulness and NIH bias is not yet

investigated in the literature. Moreover, research on information systems and organizational behaviour argue for the need to value more contextual research (Hong et al., 2014; Johns, 2006). Whereas developers' perception was previously addressed in the literature, resulting in conflicting outcomes, the cognitive state has not yet been taken into account. Thus, this research addresses the above research gaps.

3 Research Model and Hypotheses

This study considers only the most reused software assets that are software components (Sojer and Henkel, 2010), different from the work of Mellarkod et al. (2007) that considers every software artifact like code, design artifacts or requirements specifications. As Ajzen (1991) argues the behavioural intention needs to be very specific to influence the actual usage behaviour. Therefore, this research only analyses the intention to reuse existing software components. A software component is defined as an executable units of code that provides an interface, which allows its functionalities to be integrated in other software products (Ravichandran and Rothenberger, 2003). In the following, the hypotheses and research model are presented.

3.1 Performance Expectancy

Performance expectancy is defined as the "degree to which an individual believes that using the system will help him or her to attain gains in job performance" (Venkatesh et al., 2012 p .159). This construct is one the strongest predictors of intention. Usually, developers reuse existing assets to increase efficiency in their work (Sojer and Henkel, 2010). Using external software components, must create a relative advantage for developers, otherwise developers would implement everything from scratch. As using existing software components, also affects the quality of the end software product, developers would use existing components only if they have a high quality and could lower the development time or costs. In line with UTAUT, I posit the following hypothesis:

H_1: High performance expectancy is positively related to the intention to use existing software components when developing new software products.

3.2 Effort Expectancy

Effort expectancy is defined as "the degree of ease associated with the use of existing software assets" (Venkatesh et al., 2003 p. 450). Effort expectancy is an important factor within software development, as it requires not only learning and using the available API (Application Programming Interface) of software components, but sometimes it requires further modifications in order to fit the desired requirements. Thus, developers are more willing to use software components that require a low degree of effort, which could be due to a well implemented, designed and documented software component. Thus, hypothesis below can be stated:

H_2: Low effort expectancy is positively related to the intention to use existing software components when developing new software products.

3.3 Facilitating Conditions

The term facilitating conditions describes the degree to which a software developer believes that technical infrastructure exists to support the use of existing software assets when developing new software. Research on software reuse within organizations has argued that one of the success factors of software reuse is the availability of a reuse infrastructure (Sherif et al., 2006). As pointed out also by Venkatesh et al. (2012), context can have a significant influence on the result of the overall model. Thus, facilitating conditions is conceptualized to reflect the easiness of integrating an existing software component.

As such facilitating conditions will have an influence on the reuse intention of software developers leading to the following hypothesis:

H_3: Facilitating conditions are positively related to the intention to use existing software components when developing new software products.

Additionally, facilitating conditions are also expected to lower developers' effort when integrating existing components. Therefore, I propose the following hypothesis:

H_4: Facilitating conditions are positively related to the low effort expectancy of software developers when adopting existing software components.

3.4 Social Influence

Social influence refers to the degree to which software developers perceive that others believe he or she should use existing software components (Venkatesh et al., 2003). Thus, a positive attitude of co-workers towards using existing software components might influence the intention of developers to reuse more. Mellarkod et al. (2007) did not find an effect between the social factors and the perceived usefulness of reuse but they did not test for a direct effect on intention behaviour. In line with UTAUT, the following hypothesis can be posited:

H_5: Social influence is positively related to the intention to use existing software components when developing new software products.

3.5 Not-invented-here Bias

The not-invented-here bias is known in the literature as a syndrome, relating to the negative consequences it might have for organizations. As pointed out by research (e.g. Lichtenthaler and Ernst, 2006), such deviation from the optimal behaviour occurs systematically, thus we use the term bias. The first work analysing the NIH bias is that of (Clagett, 1967), which was motivated by his own experience, where the term had already been used in practice to describe the reluctance of a technical department to use an innovation created by another department (Clagett, 1967 p.1). Katz and Allen (1982) define the NIH bias "as the tendency of a project group of stable composition to believe it possesses a monopoly of knowledge of its field, which leads it to reject new ideas from outsiders to the likely detriment of its performance" (Katz and Allen, 1982 p.8). The potential negative impact of the NIH bias "include the rejection of external ideas, the underutilisation of external knowledge acquisition and the resulting negative effect on performance" (Lichtenthaler and Ernst, 2006 p.368). This negative attitude has been analysed for internal technologies, ideas, or knowledge and it can be identified at different levels: individual, project teams or organizations. In the knowledge management discipline, the identified the antecedents of the NIH bias include cultural aspects, inappropriate incentive systems, difficulties in intra-organisational communication, and status issues (Lichtenthaler and Ernst, 2006). Although the concept of NIH bias is acknowledged and described in the literature, there has been only few empirical research on it (Lichtenthaler and Ernst, 2006).

Software development provides a hotbed for the occurrence of the NIH bias. Through software reuse, "people may feel hindered in their creativity and independence by reusing someone else's software" (Sametinger, 1997 p. 16). Thus, overestimation of own skills in a particular field could lead to an "underestimation of the quality of external skills and knowledge" (Sojer and Henkel, 2010 p. 34). Additional factors contributing to the NIH bias could be the concern for the quality of the external aritfacts, but also the fear of being perceived as unprofessional or being perceived as "integrator" rather than "builder" (Lynex and Layzell, 1998). Although previous studies have found out, that developers do not show the NIH bias (Frakes and Fox, 1995), Mellarkod et al. (2007) find that the developer´s negative perception of external software assets influences the behavioural intention. Thus, developers might tend to overvalue their work as well as to underestimate the benefits of external software com-

ponents, and would therefore prefer internal development over the usage of external software components even when the last is desirable. Therefore, I propose the following:

H_6: The not-invented-here bias is negatively related to the intention to use existing software components when developing new software products.

3.6 Mindfulness

Mindfulness has been shown to reduce uncertainty and have an overall positive effect on adoption intention concerning the adoption of wikis (Sun, 2011). Mindful developers will focus more in understanding the software components and the value that these artifacts can contribute. Thus, the more mindful the developers are, the better will they be able to estimate the effort needed to solve the problem at hand and estimate the relative advantage of using external software components. In this way, they will better understand the design features of the technology, which will positively influence the adoption intention. Additionally, mindful developers will be more willing to look for alternative solutions, spending more time with the search and evaluation of external components. Thus, I can state the following hypothesis:

H_7: Software developers' mindfulness is positively related to the intention to use existing software components when developing new software products.

The antecedent of the NIH bias, such as following routines and rigid roles or the lack of incentives would be less likely to influence a mindful individual, which by definition spends more time understanding the requirements. Furthermore, mindful users are more prone to explore the design space of possible solutions based on their needs. Therefore, with respect to the NIH bias, it can be argued that mindful individuals will focus on the requirements, costs, performance and design of the software components rather than its origin, leading to a low NIH bias. Thus, I deduce the following hypothesis:

H_8: Software developers' mindfulness is negatively related to the not-invented-here bias when adopting existing software components.

Moreover, Fiol and O'Connor (2003) have argued that mindful individuals are less likely to follow the bandwagon effect. Therefore, in a mindful state, developers will be more focused on the present problem at hand and less preoccupied with the social norms. Therefore, I can posit the following:

H_9: Software developers' mindfulness is negatively related to the social influence when adopting existing software components.

3.7 Technical Expertise

Additionally to the above hypotheses, the self-reported expertise is also considered as a control variable. In open source context, Sojer and Henkel (2010) show that experienced developers reuse more code. Nevertheless, the Desouza et al. (2006) has found that experienced developers will reuse less, as it is less costly for them to code the desired artifact, than the cost related to searching and finding the right one. Moreover, as developers are experienced, they would just recall existing software components from their experience and will be reluctant to search for new artifacts elsewhere. This is due to the belief that, if the right component existed, the experience developers would most probably know about it. Thus, as developers have more expertise, the advantages of using external software components rather than developing them themselves, decreases. From this follows:

H_{10}: Software developers' expertise will be negatively related to the intention to use existing software components when developing new software products.

3.8 Research Model

The above hypotheses are integrated in the following research model.

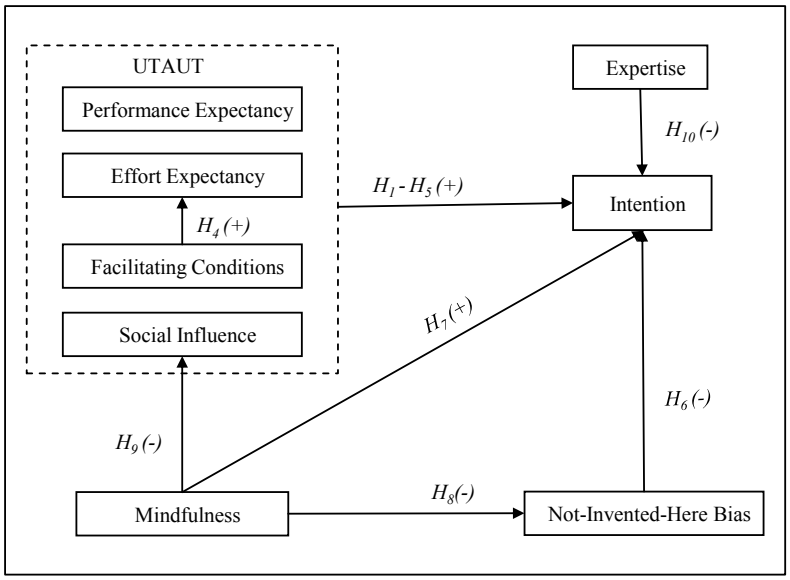

Figure 1. Research Model

4 Research Methodology

4.1 Data Collection and Sample

In order to test the hypotheses, an online questionnaire targeting software developers was developed. At the beginning of the survey, the definition and examples of software components were introduced. After several pretests with researchers and professional experts, the survey was distributed to software developers by posting the link on different blogs for professional software developers. As an incentive the users had the possibility to win one of 3 of giftcards and a tablet computer. A total of 296 responses were collected. After removing incomplete answers and those who took little time to finish the questionnaire, 142 answers were considered in the analyses. The survey was administered between May 2014 and July 2014. As expected, the data consisted of mainly male participants, which constituted 93.7%. The level of the reported education was: 37.3 % with a master degree, 31.7%, with a bachelor degree, 12.6 % with an associate degree or some college level courses, 7.7 % with a doctorate degree and the rest (10.6%) had a high-school diploma. Most of the participants were in their 20s (50.7%), 30s (30.7%) and 40s (7%). Moreover, 71.1% of the respondents were employed in an organisation, with the rest working on a freelance base.

4.2 Operationalization of Constructs

Most of the measurements were operationalized based on a multi-item and a 7-point Likert scale. Measurement items for the constructs were adapted from previous related studies (see Table 1). The items were rephrased to fit the context of the study. Most importantly, the mindfulness construct considered the openness to novelty, looking for alternatives and evaluation of the components based on the actual requirements dimensions. To operationalize the NIH bias existing items from the knowledge reuse literature were adapted to fit the context.

Constructs	No. of Indicators	Adapted from
Performance Expectancy (PE)	5	Venkatesh et al. (2012)
Effort Expectancy (EE)	3	Venkatesh et al. (2012)
Facilitating Conditions (FC)	3	Mellarkod et al. (2007)
Social Influence (SI)	3	Venkatesh et al. (2012)
Not-invented-here Bias (NIH)	4	Kathoefer and Leker (2012), (Mellarkod et al., 2007)
Mindfulness (MM)	5	Haigh et al. (2011), Sun and Fang (2010)
Intention to Reuse Software (IRS)	2	Venkatesh et al. (2012)
Expertise (self-assessment)	1	Sojer and Henkel (2010)

Table 1. Operationalization of Constructs

4.3 Instrument Validation

Before testing our hypotheses, the reliability and validity of the reflective measurement model was assessed. Content validity was established through the adoption of constructs that were used in the former studies (see Table 1) and through the initial pretest with software developers. The reflective measurements were validated as suggested by the literature (Chin, 1998). First, looking at Cronbach's alpha, which should be greater than the critical value of .70, internal consistency is assumed (Chin, 1998). Further, composite reliability was established as all the constructs are above the desired value of greater than 0 .70. Additionally, the measurement item loadings on their constructs are greater than the threshold of 0.70. The values of the average variance extracted (AVE) for all measurements are well above the accepted limit of 0.50. Therefore, the convergent validity of the measurements is assumed (see Table 2).

	AVE	C.R.	C.A.	PU	EE	SI	FC	NIH	MM	IRS
PU	0.76	0.94	0.92	0.76*						
EE	0.75	0.92	0.89	0.50	0.7536*					
SI	0.83	0.94	0.90	0.57	0.1625	0.83*				
FC	0.80	0.92	0.88	0.21	0.2641	0.11	0.80*			
NIH	0.70	0.90	0.86	-0.51	-0.35	-0.46	-0.17	0.70*		
MM	0.70	0.92	0.89	0.48	0.2331	0.32	0.17	-0.27	0.70*	
IRS	0.91	0.96	0.91	0.69	0.4137	0.58	0.21	-0.55	0.50	0.91*
C.R.: Composite Reliability; C.A.: Cronbach's Alpha; AVE: Average Variance Extracted * : Values of the average variance extracted (AVE)										

Table 2. Instrument Validation

Discriminant validity is also assumed as for all constructs the indicator loadings are higher than all of their cross loadings. Moreover, the Former-Lacker criterion, which requires that the AVE of each latent construct should be higher than the construct's highest squared correlation with any other latent constructs (Hair et al., 2011), is fulfilled by our data (see Table 2).

5 Empirical Analysis

To test the proposed hypotheses, the collected data was analysed using structural equation modelling. The software SmartPLS 2.0.M3 (Ringle et al., 2005), based on the partial-least-squares (PLS) algorithm, was used for this analysis. With SmartPLS no further sample distribution assumptions are necessary (Lohmöller, 1989). In this case, the software was used to calculate path coefficients and to determine the paths' significance in the model using the bootstrapping function. The results of the analysis are presented in Figure 2.

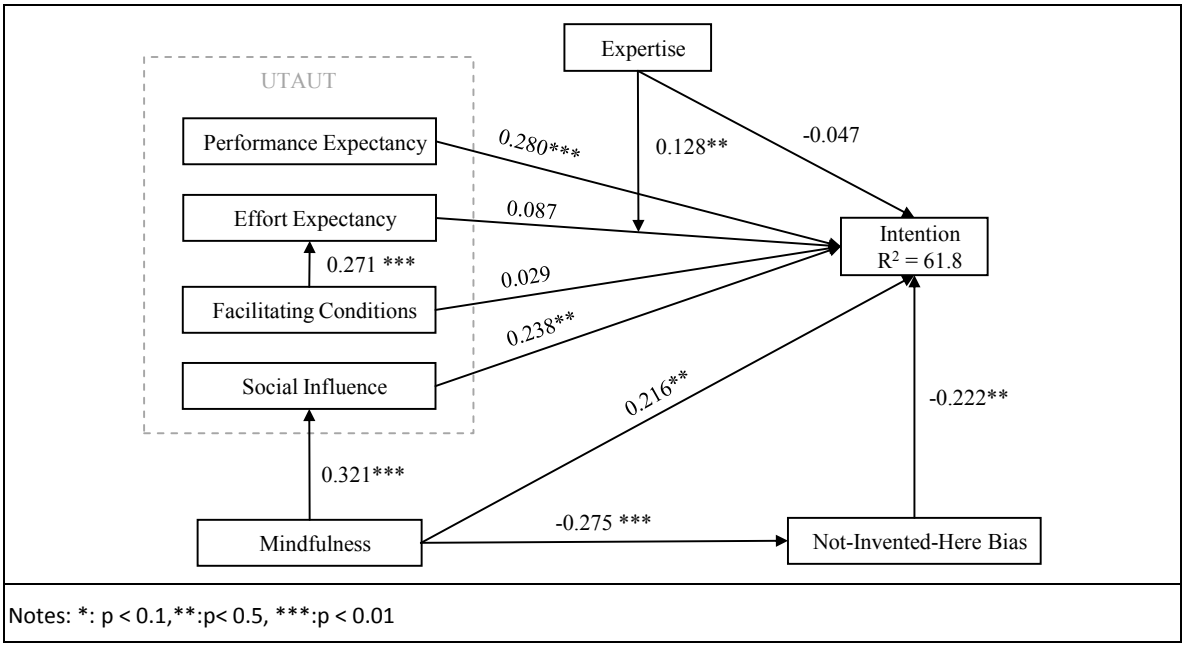

Figure 2. Results

Overall, the structural model explains a large amount of the variance of the dependent variable (R^2 = 61,8%). Consistent with the theory, the regression analysis shows that performance expectancy is the most significant factor, in explaining the adoption of existing software components, supporting the first hypothesis (β = 0,280, t=3,068). Different than expected, effort expectancy is not significant in the basic model. Only after controlling for the expertise, the interaction effect is significant, rejecting thus hypothesis H_2. Therefore, the effort perception regarding the use of existing components is an important determinant only when developers report to have a high software development expertise. The facilitating conditions did not have an influence in the decision, rejecting the third hypothesis. Nevertheless, the facilitating conditions have a strong significant influence on effort expectancy (β = 0,271, t=3,886). Social influence, on the other hand, does have a significant effect on the intention to use existing software components, confirming hypothesis H_5 (β = 0,238, t= 3,256). Similarly, the NIH bias does have a negative influence as predicted, thus confirming hypothesis H_6 (β = -0,222, t=2,516). The results show that a mindfulness state contributes positively to the intention behaviour supporting hypothesis H_7 (β = 0,216, t=2,238) and negatively influences the NIH bias, and thus providing support for hypothesis H_8 (β = -0,275, t=4,307). Differently than expected, mindfulness does have a positive effect on social influence (β = 0,216, t=2,632), leading to the rejection of hypothesis H_9. The effect of technical expertise is not significant in the model, although the sign is negative as suggested in the hypothesis H_{10}.

6 Discussion and Conclusions

The present study investigates software components reuse and extends the UTAUT model. Unlike the typical consumer or employees in an organisation employing IT, our study considered users with knowledge in software development. The study highlights the fact that individual perceptions and cognitive states are important determinants of the software developer's behavioural intention.

The results of the quantitative analysis show that performance expectancy is the strongest significant determinant of the developers' intention to use existing software components. This finding is supported by the theory and is also in accordance with previous research (Mellarkod et al., 2007). Previous studies within the software developers' context have also found that *ease-of-use* is not a direct determinant of the adoption behaviour (Mellarkod et al., 2007; Riemenschneider et al., 2002). Thus, Riemenschneider et al. (2002) argues that when the domain changes from tool adoption to the adoption of methodologies, ease of use plays a less important role. However, the results suggest that the self-reported expertise moderates the effect between effort expectancy and the software developer's intention to use existing components. This finding indicates that only for experts, low effort expectations have an impact on the software components adoption intention. One explanation for such a finding could be that for experts it's advantageous to use external software components when they are easy to use, as otherwise they might be more efficient by implementing the needed functionality themselves. Desouza et al. (2006) argue that experts usually search in their private knowledge space when they reuse software. Thus, another explanation could be that experts might end up reusing components that they are familiar with, resulting in lower effort expectations. Therefore, further research is needed to explore if such findings could be replicated in different settings. The facilitating conditions do not play a direct role in the behavioural intention, but they contribute to lower effort expectancy. The second strongest predictor is the social influence construct, which is a direct determinant even in such a specialized context as software development. Previous research has found out that developers rely on interpersonal connection when assessing the performance and the usefulness of existing software components (Ayala et al., 2011; Desouza et al., 2006). This finding can enlighten the fact that even when controlling for a mindful state, social influence has still a positive significant effect. Therefore, the opinion of others co-workers, is an important determinant for fostering the adoption of new technologies within software developers.

The NIH bias turned out to have a negative effect on the behavioural intention. This finding suggests that decision making is affected by non-rational factors, deviating from the traditional rational assumption of acceptance models. Thus, research in technology acceptance needs to actively account for such deviations (known as biases) depending on the context, and it should try to better understand its antecedents. One debasing strategy that could be adopted is the enhancement of the mindfulness state. To a limited extent, the construct of mindfulness, which describes a state in which the user makes informed decisions and does not follow routinized actions, lowers the NIH bias. Thus, the results show that a mindfulness state has a significant negative influence on the NIH bias. In addition to the negative influence on the NIH bias, a mindfulness approach positively contributes to the adoption of existing software components. Although mindfulness has a positive influence on social influence, which might suggest that mindful developers are not focused on the problem at hand, research on software reuse (e.g. Ayala et al., 2011) suggests that developers rely on the experiences of peers when making adoption decisions. One can argue that social influence contributes to a better and more efficient evaluation of existing software components.

Theoretically, this paper contextualizes the UTAUT model within the software developer's population by addressing the individual differences. The model was extended by including the cognitive state of mindfulness and the role of the NIH bias. The given research model explains a large amount of the developers' intention of adopting existing components, suggesting that the cognitive state and psychological factors are important determinants to be considered in IS adoption research. The findings show that mindfulness negatively influences the NIH bias, although to a limited extent As suggested by

Swanson and Ramiller (2004, p.560) mindfulness "entails vigilance against the proverbial not-invented-here syndrome, a response to innovations of external origin, which occurs like an antibody to preserve routine and "protect" the firm from new ideas". A mindfulness decision can not only help to mitigate the bandwagon effect (Fiol and O'Connor, 2003) or information overload (Wolf et al., 2011), but could also help to lessen the effect of the NIH bias. One of the negative consequences of the NIH bias, as previously discussed, has been the lower performance of organizations when adopting external knowledge. Therefore, by lowering NIH bias, mindfulness indirectly contributes to a better performance. Mindfulness behaviour also directly contributes to the developer's performance, as using existing software components could speed up the software development process. By adopting a mindfulness lens we are able to provide further understanding on the decision making as well as the beneficial effect of mindfulness behaviours in making sound adoption decisions. Another, contribution of this research is the evaluation of the NIH bias quantitatively when adoption external components. Thus we do see a tendency of software developers to prefer implementing the software themselves rather than reuse.

For software organisations, this research provides insights on how they can implement software reuse within the organisation. This study suggests that mindfulness can help developers to find value in the reuse of existing software components, and that it negatively contributes to the NIH bias. Therefore, managers can enhance the adoption of existing software components and lower the effect of NIH bias by enhancing mindfulness. One way to achieve this is to provide adequate, up-to-date and easily accessible information when introducing a code repository. It is also important to have different alternative solutions, so that the different technological trade-offs could be better estimated. Furthermore, the negative effect of the NIH bias can be mitigated by adopting an open organisational culture that values the use of existing components when they provide a relative advantage.

This study is one of the first that considers the effect of the NIH bias and the mindfulness concept within the software developers reuse intention. Nevertheless, this study also contains some limitations. First, the data is self-reported. Secondly, although the sample size is acceptable for IS research, a broader sample size would be helpful to further ground the model. Thus, the statistical power analysis (Faul et al. 2009) shows that test for significance can only detect medium effects, and a larger sample would be required to detect small effects.Third, the results of the study have not accounted for a specific application type, which could provide further insights. Therefore, future research needs to account for further control variables and other psychological determinants that could have an impact on the intention decision.

References

Ajzen, I. (1991). The theory of planned behavior. Organizational Behavior and Human Decision Processes, 50 (2), 179-211.

Ajzen, I. and Fishbein, M. (1980). Understanding attitudes and predicting social behaviour. Prentice-Hall, NJ, USA.

Ayala, C., Hauge, Ø., Conradi, R., Franch, X. and Li, J. (2011). Selection of third party software in off-the-shelf-based software development—An interview study with industrial practitioners. Journal of Systems and Software, 84 (4), 620-637.

Biggerstaff, T. and Richter, C. (1987). Reusability framework, assessment, and directions. IEEE Software, 4 (2), 41-49.

Boehm, B. (1999). Managing software productivity and reuse. Computer, 32 (9), 111-113.

Butler, B.S. and Gray, P.H. (2006). Reliability, mindfulness, and information systems. MIS Quarterly, 30 (2), 211-224.

Chapman, M. and van der Merwe, A. (2008). Contemplating systematic software reuse in a project-centric company. Proceedings of the 2008 Annual Research Conference of the SAICSIT, 16-26 ACM, Wilderness, South Africa.

Chin, W.W. (1998) The partial least squares approach for structural equation modeling, In Modern Methods for Business Research (Eds, Marcoulides, G.) Lawrence Erlbaum Associates, Mahwah, NJ, pp. 295-336.

Clagett, R.P. (1967) Receptivity to innovation - Overcoming N.I.H., Sloan School of Management, Master Thesis, MIT.

Dane, E. (2011). Paying attention to mindfulness and its effects on task performance in the workplace. Journal of Management, 37 (4), 997-1018.

Davis, F.D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Quarterly, 13 (3), 319-340.

Desouza, K.C., Awazu, Y. and Tiwana, A. (2006). Four dynamics for bringing use back into software reuse. Communications of the ACM, 49 (1), 96-100.

Fichman, R.G. (2004). Going beyond the dominant paradigm for information technology innovation research: Emerging concepts and methods. Journal of the Association for Information Systems, 5 (8), 314-355.

Fichman, R.G. and Kemerer, C.F. (2001). Incentive compatibility and systematic software reuse. Journal of Systems and Software, 57 (1), 45-60.

Fiol, C.M. and O'Connor, E.J. (2003). Waking up! Mindfulness in the face of bandwagons. Academy of Management Review, 28 (1), 54-70.

Frakes, W.B. and Fox, C.J. (1995). sixteen questions about software reuse. Communications of the ACM, 38 (6), 75-87.

Frakes, W.B. and Isoda, S. (1994). Success factors of systematic reuse. Software, IEEE, 11 (5), 14-19.

Frakes, W.B. and Kang, K. (2005). Software reuse research: Status and future. Software Engineering, IEEE Transactions on, 31 (7), 529-536.

Goswami, S., Teo, H.-H. and Chan, H.C. (2009). Decision-maker mindfulness in it adoption: The role of informed culture and individual personality. Proceeding of the Thirtieth International Conference on Information Systems (ICIS), Arizona, USA.

Griss, M.L. (1993). Software reuse: From library to factory. IBM Systems Journal, 32 (4), 548-566.

Haefliger, S., Von Krogh, G. and Spaeth, S. (2008). Code reuse in open source software. Management Science, 54 (1), 180-193.

Haigh, E.A., Moore, M.T., Kashdan, T.B. and Fresco, D.M. (2011). Examination of the factor structure and concurrent validity of the Langer Mindfulness/Mindlessness Scale. Assessment, 18 (1), 11-26.

Hair, J.F., Ringle, C.M. and Sarstedt, M. (2011). PLS-SEM: Indeed a silver bullet. The Journal of Marketing Theory and Practice, 19 (2), 139-152.

Heinemann, L., Deissenboeck, F., Gleirscher, M., Hummel, B. and Irlbeck, M. (2011) On the extent and nature of software reuse in open source Java projects, In Top productivity through software reuse Springer, Berlin Heidelberg, pp. 207-222.

Hong, W., Chan, F.K.Y., Thong, J.Y.L., Chasalow, L.C. and Dhillon, G. (2014). A framework and guidelines for context-specific theorizing in Information Systems research. Information Systems Research, 25 (1), 111-136.

Johns, G. (2006). The essential impact of context on organizational behavior. Academy of Management Review, 31 (2), 386-408.

Kathoefer, D.G. and Leker, J. (2012). Knowledge transfer in academia: An exploratory study on the not-invented-here syndrome. The Journal of Technology Transfer, 37 (5), 658-675.

Katz, R. and Allen, T.J. (1982). Investigating the Not-Invented-Here (NIH) syndrome: A look at the performance, tenure, and communication patterns of 50 R&D project groups. R&D Management, 12 (1), 7-20.

Kim, Y. and Stohr, E.A. (1998). Software reuse: survey and research directions. Journal of Management Information Systems, 14 (4), 113-147.

Langer, E.J. (1989). Minding matters: The consequences of mindlessness-mindfulness. Advances in experimental social psychology, 22 (12), 137-173.

Lichtenthaler, U. and Ernst, H. (2006). Attitudes to externally organising knowledge management tasks: A review, reconsideration and extension of the NIH syndrome. R&D Management, 36 (4), 367-386.

Lohmöller, J.-B. (1989). Latent variable path modeling with partial least squares. Physica-Verlag Heidelberg.

Lynex, A. and Layzell, P. (1998). Organisational considerations for software reuse. Annals of Software Engineering, 5 (1), 105-124.

McElroy, J.C., Hendrickson, A.R., Townsend, A.M. and DeMarie, S.M. (2007). Dispositional factors in internet use: personality versus cognitive style. MIS Quarterly, 31 (4), 809-820.

Mellarkod, V., Appan, R., Jones, D.R. and Sherif, K. (2007). A multi-level analysis of factors affecting software developers' intention to reuse software assets: An empirical investigation. Information & Management, 44 (7), 613-625.

Pardue, J.H. and Landry, J.P. (2009). Using Trust-TAM to explain software component adoption. Proceedings of the Fifteenth Americas Conference on Information Systems (AMCIS), 177, San Francisco, USA.

Polančič, G., Heričko, M. and Pavlič, L. (2011). Developers' perceptions of object-oriented frameworks–An investigation into the impact of technological and individual characteristics. Computers in Human Behavior, 27 (2), 730-740.

Ravichandran, T. and Rothenberger, M.A. (2003). Software reuse strategies and component markets. Communications of the ACM, 46 (8), 109-114.

Riemenschneider, C.K., Hardgrave, B.C. and Davis, F.D. (2002). Explaining software developer acceptance of methodologies: a comparison of five theoretical models. IEEE Transactions on Software Engineering, 28 (12), 1135-1145.

Ringle, C.M., Wende, S. and Will, A. (2005). SmartPLS 2.0.M3. http://www.smartpls.de, December 2013.

Sametinger, J. (1997). Software engineering with reusable components. Springer-Verlag New York Incorporated.

Sauer, C., Gemino, A. and Reich, B.H. (2007). The impact of size and volatility on IT project performance. Communication of the ACM, 50 (11), 79-84.

Schmidt, D.C. and Buschmann, F. (2003). Patterns, frameworks, and middleware: their synergistic relationships. Proceedings of the 25th International Conference on Software Engineering, 694-704, Portland, USA.

Sherif, K., Appan, R. and Lin, Z. (2006). Resources and incentives for the adoption of systematic software reuse. International Journal of Information Management, 26 (1), 70-80.

Sherif, K. and Vinze, A. (2003). Barriers to adoption of software reuse: a qualitative study. Information & Management, 41 (2), 159-175.

Sojer, M. and Henkel, J. (2010). Code reuse in open source software development: Quantitative evidence, drivers, and impediments. Journal of the Association for Information Systems, 11 (12), 868-901.

Sternberg, R.J. (2000). Images of mindfulness. Journal of Social Issues, 56 (1), 11-26.

Sun, H. (2011). Making sound adoption decisions: A longitudinal study of mindfulness in technology adoption and continued Use. Proceedings of the 32nd International Conference on Information Systems (ICIS), Shangai, China.

Sun, H. and Fang, Y. (2010). Toward a model of mindfulness in technology acceptance. Proceedings of the 31st International Conference on Information Systems (ICIS), St. Louis, USA.

Swanson, E.B. and Ramiller, N.C. (2004). Innovating mindfully with information technology. MIS Quarterly, 28 (4), 553-583.

The Standish Group (2013). Chaos manifesto 2013 - Think big, act small URL:http://versionone.com/assets/img/files/ChaosManifesto2013.pdf (visited on 08/30/2014).

Venkatesh, V., Morris, M.G., Davis, G.B. and Davis, F.D. (2003). User acceptance of information technology: Toward a unified view. MIS Quarterly, 27 (3), 425-478.

Venkatesh, V., Thong, J. and Xu, X. (2012). Consumer acceptance and use of information technology: extending the unified theory of acceptance and use of technology. MIS Quarterly, 36 (1), 157-178.

Vidgen, R. and Wang, X. (2009). Coevolving systems and the organization of agile software development. Information Systems Research, 20 (3), 355-376.

Weick, K. and Sutcliffe, K. (2001). Managing the unexpected: Assuring high performance in an age of uncertainty. Jossey-Bass San Francisco, CA.

Wolf, M., Pintner, T. and Beck, R. (2011). Individual mindfulness and IT Systems use-mitigating negative consequences of information overload. European Conference on Information Systems (ECIS), Helsinki, Finland.

Woon, I.M. and Kankanhalli, A. (2007). Investigation of IS professionals' intention to practise secure development of applications. International Journal of Human-computer Studies, 65 (1), 29-41.