Empirical Software Engineering manuscript No.
(will be inserted by the editor)

Gamification in Software Engineering: The Mediating Role of Developer Engagement and Job Satisfaction

Klaas-Jan Stol · Mario Schaarschmidt · Shelly Goldblit

Received: November 2020. Revised July 2021. Accepted: October 2021

Abstract Gamification seeks to encourage behavior of participants by borrowing elements of games, such as scoring points. Few rigorous studies exist of gamification in software organizations, and several questions have remained unanswered, for example, what might drive developers to partake, and what are the consequences of developer engagement. This article seeks to provide some answers through a rigorous empirical study at one organization that created an internal gamification platform. We develop a theoretical model that seeks to explain why developers may participate, and develop the concept of developer engagement, which we link to job satisfaction. We collected data from two sources that were linked together: developer opinion data collected through a survey, and data from the organization's version control system. We test our theoretical model using structural equation modeling and moderation analysis, and find support for our model. These findings suggest that gamification can be an effective mechanism to engage developers within the organization, and that developer engagement is positively associated with job satisfaction, which is a key outcome that is of great interest to software organizations.

Keywords Gamification, developer engagement, behavioral software engineering, job satisfaction, structural equation modeling

This work was supported, in part, by Science Foundation Ireland grant 15/SIRG/3293 and 13/RC/2094_P2 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero—the Irish Software Research Centre (www.lero.ie).

K. Stol
Lero—the Irish Software Research Centre
School of Computer Science and Information Technology
University College Cork
k.stol@ucc.ie

1 Introduction

Perhaps the most valuable resource of a software organization is its developers: it is the people within the organization that design, create, develop, and ship the software. Ensuring that developers are satisfied in their job is important, as this is a strong predictor of an intention to leave the organization (Sharma and Stol, 2020). Understanding what increases developers' job satisfaction is therefore useful to organizations. Besides keeping developers happy, organizations may wish to encourage certain types of behavior, for example, sharing code and knowledge. One potential tactic to influence people's behavior is gamification, a technique that is applicable in many different contexts and which has also been explored in software engineering (Dubois and Tamburrelli, 2013; Fraser, 2017; Dal Sasso et al., 2017). The Oxford English Dictionary (www.oed.com) defines gamification as follows:

> The action or process of making something into or like a game; spec. the application of elements of game playing (such as point scoring, competition with others, etc.) to other areas of activity, typically to encourage engagement with a product or service.

This suggests that by gamifying a process, participants' engagement may increase. While the phrase "engagement with a product or service" in the definition above might imply that participants are end-users or customers, in a software development context these participants are software developers.

The term gamification was first coined in 2002 by Nick Pelling, a game developer (Marczewski, 2013, p. 24). Studies of gamification in software engineering literature tend to focus on gamification techniques in software products (gamified software), gamification of the development process, or gamification in software engineering education. In all three strands of research, the focus tends to be on increased *engagement*. For gamified software this means increased engagement of end-users, and as suggested above, for gamification of the development process this means increased engagement of developers.

In this article we focus on gamification of the development process. We observed a number of shortcomings in the gamification literature. First, few rigorous empirical studies have focused on gamification mechanisms in industry contexts with professional developers, with a few notable exceptions (Herranz et al., 2018; Snipes et al., 2014; Neto et al., 2018; Marques et al., 2020). Many papers in this domain propose conceptual frameworks or new tools, some of which have been empirically evaluated. Several studies have been conducted with students, and while qualitative findings suggest that gamification techniques have positive effects in terms of sound software engineering practices, many of the quantitative studies have not found conclusive evidence in favor of gamification in software engineering. We suggest that one potential reason for this is that prior quantitative studies have focused on analyzing direct relationships between variables, whereas these relationships may be much more complicated in reality. For example, relationships between variables may be mediated by a third variable which acts as a 'mechanism' that explains how the criterion and outcome variable (e.g. quality of code or documentation) are related. In this study, we investigate the link between gamification participation and job satisfaction, and posit that developer engagement is such a mediating mechanism.

While several studies have suggested that gamification links to an increase in engagement of developers, the term "developer engagement" has not been clearly defined in

the software engineering literature. Thus, this is a second shortcoming in the software engineering literature. Without a clear definition, it is challenging to interpret what "increased engagement" means as it pertains to software developers. The use of the term engagement is often conflated with 'participation,' but the latter is a distinct concept and mixing these terms hinders precise measurement. Based on prior literature, we propose a definition that can be used in our study of developer engagement.

Third, we lack an understanding of *why* software developers might participate in gamification initiatives. Much of the prior work in this area has investigated whether gamification of software development tasks can lead to outcomes such as better quality of code or documentation (Dubois and Tamburrelli, 2013), developer motivation (Herranz et al., 2015), or adoption of tools (Snipes et al., 2014). Numerous studies also have studied motivational affordances, such as points and badges (Hamari et al., 2014). Such motivational affordances, however, exist within an existing gamification initiative—that is, participants are already active at that point. To the best of our knowledge, very little research has focused on what might draw software developers to a gamification initiative in the first place. Based on prior literature and Personal Investment theory (Beecham et al., 2008; Dubois and Tamburrelli, 2013; Braskamp, 2009), we suggest that developers seeking to learn new skills and technologies are drawn to gamification challenges that offer such learning opportunities.

Fourth, while engagement is typically considered as a positive outcome, one might wonder what benefits having engaged software developers offers to software organizations. Very little research indeed has focused on the consequences of having engaged developers (Alhammad and Moreno, 2020). What positive outcomes might an organization expect from engaged developers? In this article, we theorize that gamification may help to achieve an outcome that is of particular interest to software organizations: job satisfaction (Sharma and Stol, 2020). We study the relationship between developer engagement and job satisfaction, the latter being a well-known predictor of employees' intention to stay in their job.

We conducted a firm-level survey study at one organization that had started a gamification platform to encourage developers to start collaborating across teams. To focus our study, we developed a theoretical model (see Sec. 3) grounded in prior literature, which we subsequently sought to test using structural equation modeling (Sec. 4). To that end, we collected sample data from two sources, namely a developer survey and the company's GitLab installation. The results of our analysis provide empirical support for our model (Sec. 5). We find that developers participate in the company's gamification challenges to learn new skills and technologies, but that participation is moderated by the level of expertise that developers already have. Further, the results of our analysis show that participation is associated with a higher level of developer engagement. This, in turn, is positively and significantly associated with job satisfaction, an outcome that should be of great interest to software organizations (Aurum et al., 2008; França et al., 2020; Sharma and Stol, 2020). We conclude by discussing these findings and implications for practice and research.

2 Background

Deterding et al. (2011) proposed a "soundbite" definition of gamification: *the use of game design elements in non-game contexts.* Gamification seeks to change people's

behavior through increasing their motivation to act (Richter et al., 2015). It does so through motivational affordances (Hamari et al., 2014); these include giving awards, points, or badges when a person completes a task, or increasing a person's level or rank as they participate over time (Hamari et al., 2014; Richter et al., 2015). The value of points or a rank does not solely lie in *having* them, but in the implied reputation, identification with a certain group, or social approval (Deterding, 2012). The use of avatars is another technique, whereby the avatar represents the person participating in the gamified process. Gamification seeks to entice participants to exhibit certain behavior—and in a software engineering context, examples include the writing of better quality documentation, better commit messages, to writing more tests.

In the remainder of this section we summarize prior work on gamification in software engineering (Sec. 2.1). We then focus on the concept of *developer engagement*, which has remained largely undefined within the SE literature, and provide a definition (Sec. 2.2).

2.1 Gamification in Software Engineering

Whereas gamification is a topic that could be applied in any domain, in this section we focus specifically on gamification in a software engineering context. Several scholars have identified gamification as a potentially useful approach in software engineering (Yilmaz et al., 2019; Fraser, 2017; Dal Sasso et al., 2017). Numerous review papers have reported overviews of gamification in software engineering (Alhammad and Moreno, 2020; Pedreira et al., 2015; Alhammad and Moreno, 2018; Mäntylä and Smolander, 2016; García et al., 2017; Machuca-Villegas and Gasca-Hurtado, 2018); a recent tertiary study identified no fewer than 12 secondary studies (García-Mireles and Morales-Trujillo, 2019). The software engineering literature that has focused on gamification consists primarily of three strands:

- Gamification of the software product. These efforts focus on incorporating gamification elements into a product that ultimately seek to change end-user behavior (Morschheuser et al., 2018; Vargas-Enriquez et al., 2015).
- Gamification of the software development process. These efforts focus on incorporating gamification elements into the develpment process that seek to change developer behavior (García et al., 2017; Marques et al., 2020).
- Gamification of software engineering education. These efforts focus primarily on the use of gamification techniques in the teaching of software engineering to students, and as such seek to change student behavior (Alhammad and Moreno, 2018).

In this article we focus on gamification in a development process context; that is, the use of gamification to change and stimulate developer behavior. Table 1 presents prior empirical studies that have focused on gamification of the software development process, which are related to our study. That is, we selected only those papers that sought to study how gamification can help change developer behavior, excluding any paper that did not present any empirical study. We summarize and classify previous studies based on their research strategy as described by Stol and Fitzgerald (2018). We note that some studies have been conducted with students; in several cases we categorized those as field experiments, rather than laboratory experiments. The reason for this is that the experiments were conducted in a setting that was natural and pre-existing to the students, for example, when the experiment was conducted as part of a software engineering project course.

Prior reviews show that many papers on gamification present proposals for incorporating gamification in software development activities, without presenting any empirical study. Other papers present prototype tools, sometimes with an empirical evaluation. Of the papers presenting an empirical study, most were conducted with students (see Table 1) (Pedreira et al., 2015; Alhammad and Moreno, 2020; Badihi and Heydarnoori, 2017; Singer and Schneider, 2012). Relatively few papers report field studies conducted in industry (Herranz et al., 2018; Neto et al., 2018; Marques et al., 2020; Snipes et al., 2014; Passos et al., 2011). Passos et al. (2011) pilot-tested the use of role-playing game features in software task management in an industry setting, and found encouraging results. Snipes et al. (2014) applied game elements to an industry setting at ABB and found that most developers were interested in using games as a way to help them learn and improve software development practices. Marques et al. (2020) studied whether gamification contributed to adoption of Scrum practices. While their findings suggested that gamification had a positive influence on a Scrum team's 'atmosphere,' the authors did not find conclusive evidence that gamification improved to adopt Scrum practices. Alhammad and Moreno (2020) also reported a lack of evidence for the impact of gamification.

Several studies include the demonstration or propose a new tool that offers feedback, so as to stimulate certain behavior (Snipes et al., 2014; Singer and Schneider, 2012; de Melo et al., 2014; Sukale and Pfaff, 2014; Badihi and Heydarnoori, 2017). For example, Sukale and Pfaff (2014) proposed QuoDocs, a web-based system that awards points to contributors of documentation. However, no empirical evaluation was presented. Badihi and Heydarnoori (2017) developed a code summarization plug-in for Eclipse that relies on gamification mechanisms to attract contributors.

Gamification techniques have also been used to encourage developers to share knowledge. The last decade or so has witnessed the rise of social media such as Twitter and social coding platforms such as GitHub (Dabbish et al., 2012) and StackOverflow. StackOverflow is a popular Q&A platform that offers an attractive and purpose-built alternative to traditional mailing lists when it comes to sharing knowledge (Vasilescu et al., 2014). StackOverflow uses gamification techniques such as member reputation and badges (Grant and Betts, 2013; Li et al., 2012b). Gamification is closely linked to the topic of crowdsourcing, not only in a general context (von Ahn, 2006; Morschheuser et al., 2016), but also in software engineering (Stol et al., 2017b). The StackOverflow platform can be seen as a platform for crowdsourcing knowledge (Meldrum et al., 2017), and Topcoder.com is a popular platform that gamifies software development through competitions with points and monetary rewards (Stol et al., 2017a). There are numerous studies of these open platforms in the software engineering literature. Relatively few studies, however, exist on the use of gamification in software organizations. In order to understand whether gamification is an effective strategy for software organizations to pursue, rigorous research based on practical applications is necessary.

Table 1 Selection of prior empirical studies of impact of gamification on developer behavior

Study	Setting and Method	Participants	Findings
Passos et al. (2011)	Field study: pilot study to evaluate a proposed approach at a small software development company, using historical data.	13 professional developers.	Demonstrates by means of historical data the viability of gamification, but no evidence for its impact. Involved developers expressed enthusiasm about gamification mechanisms.
Prause et al. (2012)	Field experiment (i.e. in a natural setting) to measures gamifying developer reputation scores based on their improvements of maintainability and internal quality of source code through writing JavaDoc.	10 postgraduate students	No measurable improvement in quality was observed.
Singer and Schneider (2012); Singer (2013)	Field experiment: quasi-experiment to evaluate whether active feedback has an effect on commit patterns, in terms of number of commits and distribution of those over time, number and length of commit messages.	students (treatment N=37, control N=214)	Participants in the treatment group made significantly more commits, spaced out more evenly; median length of commit message was longer; more commits had a commit message.
Dubois and Tamburrelli (2013)	Field experiment to evaluate whether competition changes compliance in terms of metrics such as branch coverage, code duplication and JavaDoc documentation.	Treatment and control groups both consisting of 32 teams of 2-3 students each.	Control group only had access to their own Sonar code analysis reports; treatment group had access to all groups' reports. Treatment group exhibits slightly better scores, but results are inconclusive as no statistical tests are presented.
Snipes et al. (2014)	Field study to evaluate whether game-like feedback to the development environment would improve adoption of tools and practices for code navigation. Pre-study survey; event/usage data; post-study interviews.	Survey N=130 professional developers; logged event/usage data from a six-person team using the gamification tool	Most survey respondents (95%) would try suggested tools and practices recommended by an automated usage tracking system. Collaboration and team goals are bigger motivating factors than mandates and individual awards such as badges to try new tools and practices.
Herranz et al. (2015)	Judgment study. A pilot study to assess impact of Gamiware framework on participant motivation.	22 undergraduate students	Participants were asked to complete a questionnaire after performing a number of tasks in teams. Among the findings was that participants judged that the gamification of the tasks had increased their motivation.
Lombriser et al. (2016)	Field experiment: quasi-experiment at MaibornWolff (IT company in Germany); treatment group was exposed to 17 game elements in an online platform for requirements elicitation.	12 software developers (N=6 for both treatment and control group)	No statistical difference in stakeholder engagement between treatment and control group. Treatment group produced more user requirements which were rated higher in quality and creativity.

Continued on next page

Table 1 Selection of prior empirical studies *(continued)*

Study	Setting and Method	Participants	Findings
Yilmaz and O'Connor (2016)	Field experiment: assess whether gamification helps to improve developer motivation. Repeat administering of a survey.	30 software professionals	Results suggest that gamification increases motivation and engagement of participants.
Badihi and Heydarnoori (2017)	Judgment study: short evaluation survey of the CrowdSummarizer tool that gamifies the writing of code summaries.	149 undergraduate and postgraduate students	Participants mostly enjoyed the gamification offered by the CrowdSummarizer tool. The tool encouraged participants to write summaries (4/5). Gamification elements such as badges enhanced code summarization skills (3.4/5).
Khandelwal et al. (2017)	Laboratory experiment to assess impact of gamification on code review process.	183 undergraduate students	Experimental results indicate that gamification does not impact the code review process. Results from a post-study survey indicated that 54% of participants enjoyed the gamified process, and only 9% disliked it.
Marques et al. (2020)	Field experiment: evaluate the impact of gamification on adoption of Scrum practices by means of a custom Jira Software app.	32 professional developers ('players in app')	Results indicate no statistical difference between baseline and gamified intervention.

2.2 Developer Engagement: Untangling the Conceptual Blurring

Concepts such as customer engagement and user engagement have been well established and extensively studied in the management and business literature (Van Doorn et al., 2010; Bitrián et al., 2021). The Oxford English Dictionary's definition of gamification (see Sec. 1) explicitly refers to the encouragement of *engagement* with a product or service—clearly, this refers to customers or end-users of these products or services, rather than the creators of those products or services. In a software development context, we must interpret the term engagement to mean *developer engagement*, as it is developers who are targeted when introducing gamification of the development process. Indeed, several authors have suggested that gamification influences developer engagement (de Melo et al., 2014; Zichermann and Cunningham, 2011).

Surprisingly, the concept of developer engagement has not been explicitly defined, despite the fact that the term has been used in the software engineering literature (Adams et al., 2008; Pedreira et al., 2015; França et al., 2020; Vasilescu et al., 2014). Sukale and Pfaff (2014) presented a study design that focuses on developer engagement, no measurable definition is presented. The term engagement is often conflated with participation (Schaarschmidt et al., 2019; Adams et al., 2008), but these are distinct concepts: a developer who merely shows up for work is not necessarily an *engaged* developer. França et al. (2020, p. 129) also make this point, as they interpret the term engagement to mean *"commitment, hard-working, and interest,"* suggesting that engagement goes beyond merely showing up for work. In this study we draw a distinction between developer

engagement and developer participation. While participation is one interpretation of the term engagement, it is a narrow view on what is commonly meant by engagement in a context that focuses on employees. Whereas participation is a requirement in any work setting, engagement is only a potential energy that cannot be required in an employment contract (Suff and Reilly, 2008).

In their measurement of 'mean developer engagement,' Adams et al. (2008) equated developer engagement with participation in open source projects, using the number of projects they participated in as a proxy. We argue that counting the number of projects a developer participates in is not a measure of engagement, but only of participation, because the term engagement conveys a different connotation, as we discuss in more detail below.

This conceptual blurring has also been observed in the management literature. For example, Schaufeli (2013) noted that *"consultancy firms have conceptualized engagement by combining and relabeling existing notions, such as commitment, satisfaction, involvement, motivation, and extra-role performance."* However, each of these are distinct theoretical constructs: while a person's commitment, satisfaction, and motivation are likely to be correlated, each of these represents a different concept. Conflating these concepts makes a reliable measurement impossible, and so it is imperative to distinguish them.

As no prior definition was available, we decided to construct a definition in order to be able to apply it in this study—that is, a definition that is specifically focused on software developers and which we could measure in a quantitative study. Hence, we sought to propose a definition that has face validity within a software development context, and which is consistent with other literature on employee engagement. Intuitively, an engaged developer is a person who actively performs within an organizational context, which may be a corporation or an open source project. To define the term 'developer engagement' more precisely in a way that is measurable, we draw on prior literature to identify two key characteristics. Prior research defined engagement as: *"The degree to which workers identify with, are motivated by, and are willing to expend extra effort for their employer"* (Suff and Reilly, 2008). Devi (2009) defined engagement as *"the extent to which an employee puts discretionary effort into his or her work, beyond the required minimum to get the job done, in the form of extra time, brainpower or energy."* Perhaps most relevant in the context of software developers, Hertel et al.'s study of Linux developers (Hertel et al., 2003, p. 1168-9) found that open source participants' engagement could be predicted based on (among others) their identification as a Linux developer and their motivation to improve their programming skills. From the above, we identify two key characteristics of engaged developers. First, engaged developers exhibit a form of self-identification with the community in which they work. Second, engaged developers seek to improve their skills to help do their job. Thus, we propose the following definition of an engaged developer:

> An engaged developer (a) is part of, and identifies with, the software development community they work in, and (b) actively seeks to improve their software development skills.

This definition is consistent with other characteristics typically used in the context of engagement. For example, França et al. (2020) have used terms such as "committed," "hard-working," and "has an interest" in the work. While each of these terms (committed, hard-working, having an interest) represents a separate and distinct concept, we argue

that they are closely related to what we define as engagement, and indeed, are likely to correlate highly. Advancing one's professional software development skills aligns with what Kahn (1990) referred to as people's expression at a *cognitive* level in their work; that is, *"task behaviors that promote connections to work,"* which we interpret as developers advancing their professional development skills. Seeing oneself as a member of a developer community aligns with what Kahn (1990) described as a person's *emotional* expression; at this level, we can think of behavior that promotes connections to other people—which we link to being part of a group.

In defining developer engagement, it is worth considering what developer *disengagement* means. After all, this study seeks to measure developer engagement, and so it is expected that some respondents score low on this attribute. Following Kahn (1990, p. 694), who discussed personal disengagement at work, disengaged developers *"uncouple themselves from work roles"* and *"withdraw and defend themselves."* In other words, they are unlikely to see themselves as part of a developer community (withdrawal) or to invest in their professional skills (uncoupling from work).

It is worth noting that there is a considerable literature that seeks to define employee engagement (Macey and Schneider, 2008), and we acknowledge that different definitions present different conceptualizations of the term. As we mentioned, some existing definitions or descriptions include concepts such as job satisfaction and motivation. We sought to offer a definition that has face validity, is practical, relevant specifically to software developers (rather than a generic 'employee'), and measurable in a quantitative study. Finally, while we argue that our definition is a useful one for the purpose of this study, this does not imply that other definitions are not also possible. As is common in the scientific enterprise, different definitions (whether consistent with one another or competing) of terms are possible, and our proposed definition should not preclude other scholars to propose alternatives.

3 Theory Development

To guide and focus our study we develop a theoretical model. We build our model drawing on several established theories and salient themes from prior work on gamification in software engineering.

3.1 Self-Improvement and Participation in Gamified Tasks

Our first question focuses on *why* developers might participate; what is their motivation to partake in a gamification platform? Numerous theories of motivation exist and developer motivation is a well-studied area within the SE literature (Beecham et al., 2008; França et al., 2020); Rapp et al. (2019) lamented a lack of theoretical diversity in the gamification literature, noting that self-determination theory (SDT) is referred to most frequently. SDT suggests that people have certain psychological needs that need to be satisfied that drive motivation, specifically, autonomy, competence, and relatedness. While these three factors of SDT offer a comprehensive view to explain motivation, in this study we seek to focus specifically on the question what might draw developers to a gamification platform. Thus, we take a pragmatist approach in selecting an alternative motivational theory that helps us understand why developers do what they do, namely, Personal

Investment (PI) theory (Braskamp, 2009). PI theory suggests that *"people invest in themselves in certain activities depending on the meaning these activities have for them"* (Maehr and Braskamp, 1986, p. 62); one important factor that explains behavior is personal goals (Braskamp, 2009). Prior research suggested that developers are more motivated when they have a desire to learn new skills (Beecham et al., 2008). Dubois and Tamburrelli (2013) also suggested that gamification may motivate developers to learn new technologies. Given the fast pace of technological advances in the software sector, it is important that developers remain pro-active, for example by learning new skills and technologies. Gamified software development challenges offer a compelling mechanisms for developers to both contribute to productivity and performance, and to learn new technologies. Hence, we propose the following hypothesis:

Hypothesis 1 (H1). Developers' desire to learn new skills and languages is positively associated with participation in gamification challenges.

A corollary of Hypothesis 1 is that the level of expertise that developers exhibit is likely to curb the inclination to participate in gamified challenges. Hence, we propose that Expertise has a negative moderating influence on H1:

Hypothesis 2 (H2). Developers' level of expertise negatively moderates the association between desire to learn and participation in gamification challenges.

3.2 The Role of Participation in Developer Engagement

Scholars have suggested that gamification can lead to more developer engagement (Herranz et al., 2015; García et al., 2017; Sukale and Pfaff, 2014). As discussed above, we defined developer engagement as a two-dimensional concept, namely, active participation and identification with a developer community, and the pro-active advancement of professional development skills. França et al. (2020) suggested that learning new technologies is a driver of developer engagement. Further, a range of factors were found to be facilitators of developer engagement, including working on challenging tasks, work variety, and learning opportunities (França et al., 2020). Several other studies found support for the link between participation in a gamified setting and engagement (Hamari et al., 2014). Further, several studies found that gamification helped to increase student engagement in a classroom setting (Cheong et al., 2013; Welbers et al., 2019). Other studies found increased levels of user engagement when gamification was used to learn complex software programs, such as AutoCAD (Li et al., 2012a) and Adobe Photoshop (Dong et al., 2012).

Overall, prior studies suggest that gamification leads to higher levels of engagement, though the engagement concept has been measured in different ways for different types of participants (users, students). To test whether gamification is linked to a higher level of *developer* engagement, we propose the following hypothesis:

Hypothesis 3 (H3). Participation in gamification challenges is positively associated with developer engagement.

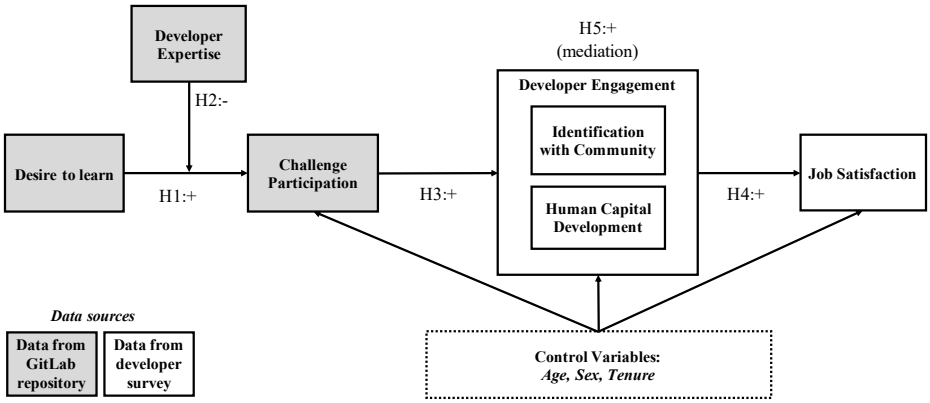

Fig. 1. Research model of this study. Developer Engagement is a second-order construct.

3.3 The Role of Developer Engagement in Job Satisfaction

Developers who are more engaged in their daily job are more likely to invest a sense of their self into the job; they *want* to succeed in their job, and they want others to succeed as well (Aurum et al., 2008), as they are more than likely dependent on them (Strode, 2016).

Job satisfaction of software developers has received considerable attention in recent years (França et al., 2020; Sharma and Stol, 2020; Lenberg et al., 2015). Evidence from psychology literature suggests a positive link between different forms of 'engagement' and job satisfaction, including work engagement (Karanika-Murray et al., 2015), organization engagement, and employee engagement (Saks, 2006). As we defined *developer engagement* as a distinct concept, an open question is therefore whether this, too, is an antecedent of job satisfaction. Hence, we propose that developers who are more "engaged" in their work exhibit a higher level of job satisfaction:

Hypothesis 4 (H4). Developer engagement is positively associated with job satisfaction.

3.4 The Mediating Role of Developer Engagement

The previous hypotheses have focused on the specific links between participation, engagement, and job satisfaction. While studying these individual links is insightful, ultimately we are seeking to identify mechanisms that lead to positive attitudes and behavior. We argue that developer engagement is such a mechanism that explains the link between participation and job satisfaction. It is not the participation itself that leads directly to job satisfaction, but rather through developer engagement. Developer engagement is thus a mediating factor in the relationship between gamification participation on the one hand, and job satisfaction on the other. It is our proposition that developer engagement is a central mechanism that "translates" participation into positive outcomes that software development organizations should strive to achieve. Hence, our last hypothesis is:

Hypothesis 5 (H5). Developer engagement mediates the association between participation in gamification challenges and job satisfaction.

4 Research Method

In order to evaluate the theoretical model, we conducted a firm-level sample study to collect data that were subsequently analyzed using structural equation modeling. As discussed in Sec. 2, gamification refers to a broad category of activities and mechanisms to mobilize people. Gamification in software development settings is therefore likely to vary considerably across organizations. Because the mechanisms and operationalization of gamification may be different in each setting, conducting sample studies across different organizations is challenging as comparisons can be difficult to perform. Hence, we conducted a sample study at one organization that sought to increase collaboration across departments and teams. As such, this study's research strategy is a hybrid between a field study conducted in a specific natural setting (seeking to capture rich details of a specific context), and a sample study (seeking generalizability to a population) (Stol and Fitzgerald, 2018). While it achieves a level of generalizability, it is limited to the context of the specific field setting. On the other side, while generalizability is limited, it offers more contextual details than would otherwise be available in a cross-sectional survey.

4.1 Background of the Research Setting

We conducted the study at XCorp (a pseudonym). In order to protect the identity of the organization, we do not report its geographic location. XCorp is a large organization with a developer workforce of approximately 850 fulltime staff distributed across several cities (within a single country). XCorp creates software solutions for the defense sector.

In Fall 2017, XCorp had started an inner source initiative (Cooper and Stol, 2018) in order to encourage more cross-team collaboration, increase software reuse and reduce code duplication, and to offer more attractive opportunities for its staff to learn new technologies. Inner source seeks to adopt open source principles in software development within organizations. However, after opening up the version control system to allow developers from different teams to collaborate and share code, very little collaboration happened for several months. To understand why, the company started to identify several reasons for this. They (informally) identified several developer 'profiles.' A first category of developers were topic experts, and were looking for opportunities to 'shine' by demonstrating their skills to others. Others were 'code junkies' who were looking for opportunities to evolve professionally. Others still were 'code ninjas'; these developers were continuously looking for new opportunities to learn new technologies, and would typically stay with the company for only a few years. Some of the more senior developers were not particularly keen to change the type of work they were doing, though they did express a mild interest in learning new technologies, preferably in a 'safe' environment so they could experiment with these that would not be scrutinized too closely by their colleagues. Other profiles included 'lone horses' who were looking for opportunities to collaborate, 'newbies' who were looking for mentorship and a way to orient themselves, and networkers who were looking for opportunities to connect to others. A final category of developers were the socially 'shy,' who sought a way to connect informally; the weekly challenges would give them a topic to discuss that was interesting and relevant to them.

To help change developers' behavior, XCorp created a gamification platform, called "The Guild" (a pseudonym), to create an internal developer community and ecosystem.

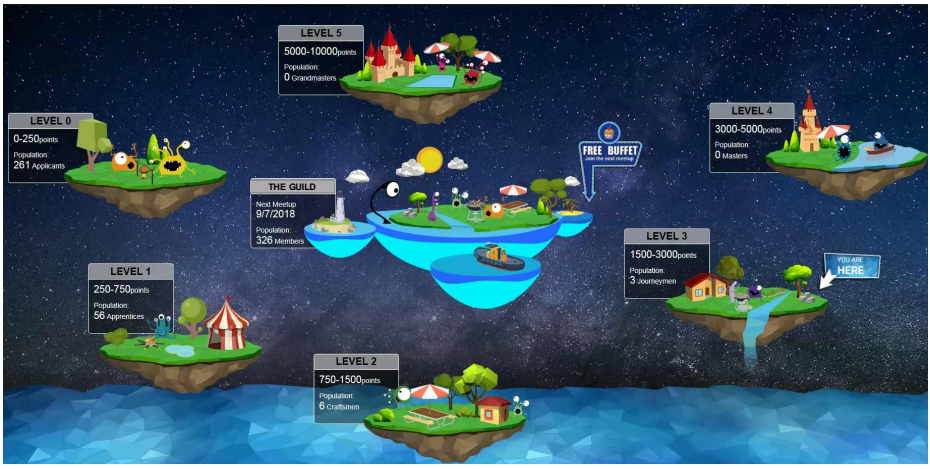

Fig. 2. Screenshot of the gamification platform

The gamification platform has a playful and attractive user interface created by a professional graphics designer (see Fig. 2). The playful design was a deliberate decision to signal that this system was like no other in the organization. XCorp has had a very strong and traditional engineering approach with considerable regulations (due to the defense domain it operates in), and all other corporate systems exhibited a typical dull design. The interface depicts "floating islands" that represent the various levels; each new person starts as an 'Applicant.' In order to be accepted to the guild, a developer would have to be nominated by an existing guild member within the platform. As developers completed more challenges and tasks, they could progress to a higher level, ultimately to the highest levels of 'Master' and 'Grandmaster.' The islands are inhabited by cartoon-style monsters (see Fig. 3). Developers can earn stickers depicting these monsters when they complete certain tasks on the platform—these stickers soon became rather popular. The first tasks are rather easy to complete (e.g., update one's profile, create or clone a repository), but as developers progress, it becomes harder to earn the stickers. The stickers are much sought after, as developers (and managers) decorate their laptop computers with them, showing off to others how active they are. Other markers of activity include points and badges.

In addition to development tasks, the platform hosts weekly interactive "challenges" to prompt developers to participate. When solved, developers earn another sticker. Challenges are created by senior community members and vary in level of complexity. The challenges are not directly related to any product deliverable; instead, they are designed specifically to attract developers to participate and encourage them to become members of the internal community. A typical example of such a challenge was around the introduction of SonarQube. While there was initially little interest from developers to invest time to learn SonarQube, when a challenge was created to answer specific questions about this package, developers started to participate in these challenges. These challenges thus prompted developers to invest time and learn new technologies and skills.

From time to time, there is an "Adopt a Monster" campaign, where members have to complete tasks to qualify as adopting 'parent.' Member adoption status is only visible in

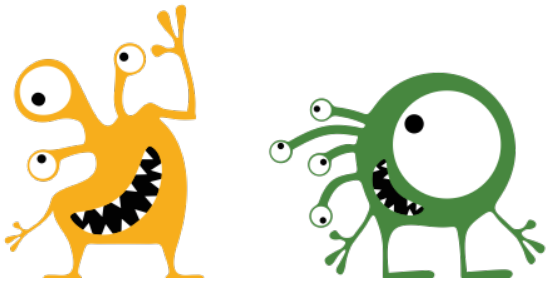

Fig. 3. Two monsters from XCorp's gamification platform

the game. The winner is selected through a lottery system, who may choose the monster's name, personality, and select an avatar representing the monster, which would live in the guild gamification platform. As these adoption opportunities are not frequent, there is considerable competition for this.

Further, physical in-person technology meet-ups are organized to discuss new technologies and challenges. These meet-ups are only accessible to members of the guild community, and so these meet-ups serve as an additional motivation to become active. Friends and colleagues from other companies may be invited on occasion as well. The platform also has a chatroom, for which developers can choose to create an account. The organization adopted a private installation of the GitLab version control system. Any activity in relation to the gamification challenges is linked to the GitLab installation.

Challenge meetings are organized in an ad-hoc fashion, when a specific need arises for a specific group within the organization. For example, one group sought to recruit moderators to their repositories by hosting a workshop of contributions, with at least one representative of each of their customer software groups. Another software team organized a challenge meeting to assist in their shift to the Git version control system. A one-day "elevator challenge" was organized. Participants could submit new elevator algorithms to compete against the elevator algorithm currently in use on the premises of XCorp—this was jokingly referred to as the least efficient elevator algorithm in the universe, reflecting the frustration of developers within the multi-storey building. To participate, developers had to learn and perform many common Git commands so as to become familiar with this system, including cloning the base repository, forking repositories, creating pull requests, and so on.

4.2 Research Design

To test the theoretical model in Fig. 1 we use structural equation modeling (SEM). The hypotheses in our theoretical model in Fig. 1 rely on various theoretical constructs. These constructs, such as 'expertise' and 'job satisfaction' are not easily or reliably observed directly, but rather constitute latent variables. Latent variables are those that cannot be directly observed, but are measured through indicators that *are* directly observable. A latent variable is a hypothesized construct that a researcher believes to exist; to measure a change in a latent variable, we measure a change in a set of observable

variables (i.e., indicators) that we believe to be caused by the underlying latent variable.[1] For example, Longo and Mura (2011) measured the construct job satisfaction by the following indicators:

- "I am satisfied with my promotion opportunities"
- "I am satisfied with the recognition I receive for a job well done"
- "I am satisfied with the amount of say I have in how my work is done" and
- "I am satisfied with my job."

These indicators, phrased here as statements, can be measured on a Likert scale.

Rather than relying on a single indicator to measure job satisfaction, using multiple indicators allows us to more reliably measure this construct. The common factor is 'job satisfaction,' and the part of an indicator that is not associated with the common factor is considered measurement error (Rigdon et al., 2017). In this case, we argue that job satisfaction comprises several facets, such as promotion opportunities and receiving recognition for a job well done. In this case, job satisfaction is modeled as a reflective construct: a change in the value of the theoretical construct job satisfaction causes a change in the indicators. This is conceptually shown in Fig. 4.

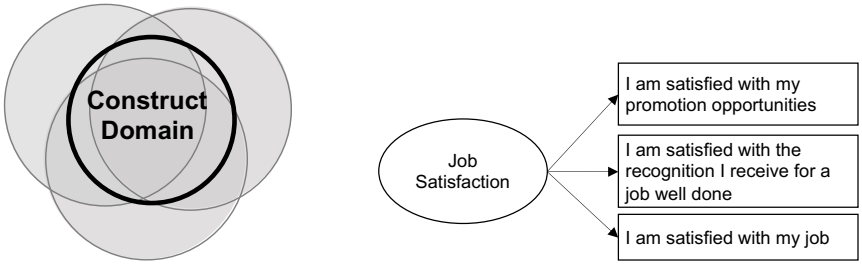

Fig. 4. Left: the reflective latent variable exists where the three indicators overlap; variance that is not caused by the latent variable is measurement error. Right: measurement of a reflective theoretical construct with multiple indicators (Adapted from Hair et al. (2016))

To assess whether the different items are consistent—that is, they change in consistent ways as the value of the latent variable changes—it is common practice to calculate Cronbach's Alpha. In exploratory research that relies on new measurement instruments, values between 0.6 and 0.7 are deemed acceptable; for more mature instruments, values over 0.7 are typically expected (Hair, Jr. et al., 2013). Values over 0.95 indicate that the indicators might be too similar, indicating redundancy, and thus such values are less desirable.

The part of the model that represents the definition of the latent variables as sets of indicators is called the measurement model. The part of the model that represents

[1] This is the case for *reflective* constructs: a change in the construct is reflected in its indicators. Constructs to study human behavior or personal attributes tend to be reflective. Constructs can also be modeled as *formative* latent variables, whereby a construct is hypothesized to be caused by a set of indicators. For example, to measure drunkenness without special instrumentation, one could count the number of pints of beer, glasses of wine and shots of spirits. Together, their intake *causes* drunkenness, not the other way around. Modeling formative constructs is, however, not without considerable problems, the primary issue being the ability to identify (i.e. estimate parameters for) a model; we refer interested readers to Bollen (2011)'s treatment of this topic.

the relationships between the latent variables is called the structural model. Structural equation modeling software calculates the whole model at the same time.[2] The next section discusses how each of the theoretical constructs was measured for this study.

4.3 Measurement Instruments

We used measurement instruments from prior literature where possible. Together, the measurement instruments are called the measurement model of the structural equation model. These manifest variables came from two separate data sources. The first three constructs (Desire to learn, Expertise, Gamification Participation) were measured using data from XCorp's GitLab installation. Data for the remaining constructs were collected through a developer survey which relied on developers self-reporting on a series of statements that were answered with 7-point Likert scales (1=Strongly disagree, 7=Strongly agree). These two datasets were subsequently joined using respondents' GitLab identifiers (also collected as part of the developer survey) into a single database. Below we describe the items (observed variables) that comprised the various constructs, and we report the Cronbach alpha values as a measure of internal consistency.

Desire to learn. To measure Desire to Learn, we used data from the GitLab repository. All employees can report in their GitLab profile the languages and skills that they would like to learn. We counted the number of languages and skills, and so the construct Desire to Learn is defined as a two-item construct, namely (1) the number of languages the person would like to learn, and (2) the number of skills the person would like to learn ($\alpha = 0.83$). Relying on this profile data from the GitLab installation means that it is more likely that this information is complete: it is very unlikely that survey participants would have made an effort to list all skills and languages they would like to learn, as this can be time consuming. Further, because this information comes from developer profile pages, we consider these sincere expressions of what developers seek to learn, and thus that these profile pages are a reliable source of information.

Expertise. We defined expertise as a two-item construct similar to Desire to Learn, but instead of relying on the languages and skills that a person would like to learn, we relied on (1) the number of languages and (2) the number of skills that person had listed (in their GitLab profile) as having knowledge of ($\alpha = 0.74$).

Gamification Participation. Participation in the gamification was operationalized with three indicators. First, the number of challenges that the person has solved. While this is clearly a good indicator of participation, not all participants in a challenge would have necessarily solved the challenge, and so we selected additional indicators. The second indicator is the number of challenge meetings that the developer has attended. Such meetings facilitate discussion with others, for example, and attending is therefore an 'indicator' of participation. The third indicator is a binary variable indicating whether a developer has a chatroom handle; developers who have a chatroom handle may interact with others to discuss challenges. Given that the three items have very different scales, we calculated the standardized Cronbach's alpha which relies on the correlations instead of covariances between items (Furr and Bacharach, 2008, p. 117), resulting in a Cronbach α of 0.81.

[2] This is one of the key differences with PLS-SEM, which calculates the 'scores' of latent variables based on their indicators first, before the structural model is calculated.

Developer Engagement. Our theoretical model focuses on the concept of developer engagement (Sec. 2). Bollen (2011, p.360) pointed out that the meaning of concepts are captured in theoretical definitions, and that, *"the theoretical definition should make clear the number of dimensions in a concept. Each dimension is represented by a single latent variable in a SEM."* Our definition of developer engagement clearly delineates two dimensions: identification with the internal developer community, and human capital development. Hence, we operationalized developer engagement as second-order latent variable that consists of these two latent variables (no Cronbach α is calculated for second-order constructs). We discuss these two latent variables below.

Identification with the Community. Identification with the Community (ID) refers to the degree to which a developer identifies themselves with the community—participating in any community typically leads to a person seeing themselves as part of that community, and as such, such a person engages with the community through interacting and potentially helping others. We used two items from Chiu et al. (2006)'s instrument to measure ID ($\alpha = 0.88$). Items included "I feel a sense of belonging towards the Guild community" and "I am proud to be a member of the Guild community."

Human Capital Development. Human capital (HC) refers to a more personal attribute, namely the extent to which a developer focuses on developing their professional skills. Again, this reflects the level of engagement that the developer has with the job at hand, and to improve one's skills. We adopted a three-item instrument to measure Human Capital Development ($\alpha = 0.94$). Items included "Participation in the Guild improves my software development performance," "Participation in the Guild community helps my professional growth by developing my skills and learning new technologies, tools, and practices," and "Participation in the Guild community advances my skills in developing software."

Job Satisfaction. We adopted a four-item instrument from Longo and Mura (2011); items included: "I am satisfied with my promotion opportunities," "I am satisfied with the recognition I receive for a job well done," "I am satisfied with the amount of say I have in how my work is done," and "I am satisfied with my job." However, given the poor result of reliability measures, we removed item 3, leaving a three-item instrument ($\alpha = 0.80$). Upon closer inspection, item 3 seems to capture an aspect that is categorically different from the others, in that it relates to *how* work is done, whereas the other items reflect on promotion, recognition, and a holistic view of the job itself.

Control variables. We included three control variables in the model: age, measured in years; tenure, measured in years, and sex, recorded as a binary variable (male=1, female=0). As mentioned, XCorp operates in the defense sector which is a more conservative context, where all employees are vetted and hold the highest security clearance. Unfortunately, this also means that the population is likely to be less diverse (mostly men, with a small proportion of women), and where people may be less likely to report if they identify otherwise.

4.4 Data Collection and Analysis

Data were collected from two separate sources. We first collected data through an internal online survey, facilitating self-reported data. The survey was sent to all developers; in order to encourage participation, we committed to make a donation of approximately US $2.00 per response to a charity that was local to XCorp. As part of the questionnaire,

we asked respondents to report their internal GitLab user identifier. In so doing, we assured respondents that data would be treated in strictest confidence. We received 155 usable responses (a response rate of approx. 18%). Using the GitLab user identifier, we collected data from the internal GitLab system for behavioral data, that is, data reflecting respondents' actual activity and goals (such as skills and languages a developer would like to learn). Table 2 reports descriptive statistics of the sample.

As briefly noted, we used structural equation modeling (SEM). SEM represents a family of second-generation statistical approaches; so-called first-generation methods include ANOVA and multiple regression, which typically rely on ordinary least squares (OLS) estimation. SEM offers a number of benefits over traditional regression analysis methods commonly used in SE research. Traditional multiple regression allows for only a single outcome variable, whereas SEM allows for any number of outcome variables. Traditional regression does not allow modeling of indirect (or mediating) effects. Further, traditional multiple regression does not support the use of latent variables—that is, only directly observed data can be used, which are assumed to be without measurement error. Many constructs of interest in SE research, however, cannot be directly observed, such as trust. SEM allows measurement of such latent constructs by means of measurement instruments, which can account for measurement error (see Sec. 4.2).

SEM is extensively used in other research fields, including Information Systems, Marketing, and Management, but rarely in software engineering research. Notable exceptions are Capra et al. (2008)'s study of software quality and governance in open source projects, Lindsjørn et al. (2016)'s study of teamwork quality and project success, and more recently Ralph et al. (2020)'s study of the impact of the COVID-19 pandemic on software developers' wellbeing.

One of the benefits of SEM over first-generation methods is that SEM allows the inclusion of latent variables as opposed to only directly observable variables. As attention for human factors and behavior in software engineering is increasing, the use of SEM is highly suitable in order to analyze variables such as trust and job satisfaction (Lenberg et al., 2015; Graziotin et al., 2020). This study therefore makes a methodological contribution to the limited few studies employing SEM by demonstrating its use in a software engineering context.

In this study we rely on covariance-based SEM (CB-SEM), as opposed to Partial-Least Squares SEM (PLS-SEM) which is used more commonly in studies published in SE venues (Russo and Stol, 2021). The two approaches to SEM rely on a different computation model and assumptions (Rigdon et al., 2017). In the remainder of this paper, we use the term SEM to mean CB-SEM. A novel aspect of our study in comparison to prior SEM studies in SE is the use of two separate data sources for some of the predictor and criterion variables. Most SEM studies rely on a single data source, typically a respondent survey. This could lead to common method variance, which refers to

Table 2 Age, tenure, sex of respondents

Variable	Min.	Mean	Max.	SD	Count (%)
Age (years)	25	38.5	63	8.3	
Tenure (years)	0.3	7.6	35	6.2	
Male					112 (72.3%)
Female					43 (27.7%)

an artificial covariance between variables due to the use of the same (common) data collection—Podsakoff et al. (2003) presents an extensive overview of this phenomenon. The use of separate data sources is a key strategy to prevent this issue. Section 4.7 discusses this issue in more detail.

We used the Lavaan library for R (v. 0.6-9) to evaluate the structural equation model, within RStudio (RStudio Team, 2020) (v. 1.2.5042) and R (v. 3.6.2). Lavaan is a mature open source library that is actively being developed and supported (Rosseel, 2012). A replication package is available on the Zenodo platform (`https://doi.org/10.5281/zenodo.5549698`).

Most SEM analysis software packages, including Lavaan, do not have native support for moderation analysis with latent variables (though it does support moderation with observed variables). For that reason, we evaluated Hypothesis 2 in a separate Lavaan model using factor scores. Factor scores combine all values of a latent variable's indicators into a single value; hence, the score represents the factor as a whole. One could potentially run the complete model with factor scores, reducing the structural equation model to a path model. However, doing so would ignore any measurement error, which is unrealistic and would bias the parameter estimates.

4.5 Model Fit Evaluation

SEM does not only evaluate the significance and strength of individual relationships between two variables, but rather has a more holistic focus on the overall fit of the model. A first step in SEM is to evaluate the measurement model; that is, to establish the validity of the measurement of the latent variables (the constructs) by means of the various indicators. Section 4.6 presents details on the measurement model. Part of the analysis in SEM includes an assessment of the model fit, which we discuss in this section.

Model fit in traditional regression models is assessed by examining the explained variance (R^2) in the dependent variable. A researcher using traditional regression seeks to assess the extent to which variables can explain a dependent (outcome) variable. This search for maximizing the explained variance is also a focus in PLS-SEM (Hair et al., 2016; Reinartz et al., 2009). However, this is not the goal in CB-SEM, which seeks to establish whether a proposed theoretical model is supported by the data. While an assessment of the explained variance is still of interest, it has no role in assessing model fit. As Kline (2016, p. 264) pointed out, *"overall model fit and R^2 for individual outcomes are basically independent. For example, disturbances in structural models with perfect fit can still be large (i.e., R^2s are low), which means that the model accurately captures the relative lack of predictive validity in the data."*

Instead of a focus on the explained variance, a covariance-based SEM model is evaluated by assessing the extent to which the proposed theoretical model 'fits' the data. If the model fit is poor, then it should be discarded: a well-fitting model is required in order to be able to interpret the estimated parameters of the model. Since CB-SEM is not commonly used in software engineering studies, we briefly discuss model evaluation next.

Structural equation models are represented by a set of matrices. The theoretical model is transformed into a *model-implied* variance-covariance matrix: it captures the variances and covariances that are encoded in the model. The input data are also transformed into a variance-covariance matrix; this is the *input* or *analyzed* variance-covariance

matrix. A SEM software package tests the null hypothesis that the model-implied and input variance-covariance matrices are statistically the same (i.e., good model fit). The discrepancy between the two is minimized using an estimator (discussed below), and generates a test statistic that follows a χ^2 distribution. The first test is thus whether the χ^2 statistic is significant or not, whereby the desired result is a nonsignificant χ^2. That is, a researcher operating within a SEM framework seeks support for the null hypothesis, namely, that the data are consistent with the theoretical model.

The difference between the model-implied and input matrices is minimized by an estimator. Several estimators are available in most SEM packages, and the normal theory-based Maximum Likelihood is the default. There is a considerable stream of methodological research into the performance of estimators under varying circumstances. The standard ML estimator assumes multivariate normality of the endogenous variables (Finney and DiStefano, 2012; Kline, 2016)(Kline, 2012, p. 122); as some of our data were not normally distributed, we used MLM, which is a robust variant of the standard ML estimator that does not rely on the assumption of normality (Gana and Broc, 2019; Savalei, 2018).[3]

In most models, the χ^2 test will yield a significant result; that is, the model-implied matrix and input matrix will differ in a statistically significant way, not least because calculation also depends on the sample size. While this is a subject of some heated debates even today, most scholars accept models that have a significant χ^2. Other, alternative 'global' measures of fit have been proposed, the most common of which are briefly discussed below (cf. Bagozzi and Dholakia (2006); Schumacker and Lomax (2016); Kline (2016); Hu and Bentler (1999)) and summarized in Table 3. These fit indices are called global because they assess the fit of the complete model, including measurement and structural model (Williams and O'Boyle Jr, 2011).

The Root Mean Square Error of Approximation (RMSEA) is a measure of how well the theoretical model fits the data. Values of up to 0.05 indicate a good fit, whereas values between 0.05-0.08 indicate an acceptable fit. SEM software packages also report a 90% confidence interval (CI) for the RMSEA; the upper limit of the CI should be smaller than 0.10 (Loehlin and Beaujean, 2017). Further, SEM packages also report the probability that the RMSEA value is below .05; sometimes this is referred to as the *p* of Close Fit, or PCLOSE (e.g. in SPSS Amos). A nonsignificant *p* of Close Fit indicates good fit (Kline, 2016); however, this test often yields a significant result as it relies on the sample size, and so even the smallest 'misspecification' is magnified by a large sample size. Most researchers in the social sciences accept a significant *p* of Close Fit.

Two other fit measures are the Comparative Fit Index (CFI) and the Tucker-Lewis Index (TLI). Desired values for both these indices are 0.95 or higher, though values slightly lower would not necessary disqualify the model. The Standardized Root Mean square Residual (SRMR) has a preferred cut-off value of 0.08, with smaller values indicating better fit. The ratio of the χ^2 and the model's degrees of freedom (df) is another frequently used measure of fit, though its use has been critiqued. In practice, it is a useful approximate indicator of model fit, though it should not be relied on exclusively. The most stringent rule of thumb suggests values smaller than two suggest a good fit (Tabachnick and Fidell, 2013, p. 720).

[3] We also ran the model with the MLMV estimator, which is similar but incorporates a mean and variance correction of the calculation of chi-squares (Savalei, 2018). The results were similar and consistent with MLM.

Table 3 Common fit measures for evaluating structural equation models

Fit measure	Desired value
p-value χ^2	Nonsignificant result ($p > 0.05$) indicates good fit[1] (Kline, 2016).
Root Mean Square Error of Approximation (RMSEA)	< 0.05 indicates close fit; values 0.05–0.08 indicate good to acceptable fit (Hu and Bentler, 1999; Schumacker and Lomax, 2016; Kline, 2016).
90% confidence interval RMSEA	Upper limit of CI should be < 0.10 (Loehlin and Beaujean, 2017, p. 71)
p-value RMSEA $< .05$ (p of Close Fit, PCLOSE)	Nonsignificant result ($p > 0.05$) indicate that the data fit the model well (Kline, 2016).[1]
Comparative Fit Index (CFI)	≥ 0.95 indicates good fit (Hu and Bentler, 1999; Schumacker and Lomax, 2016)
Tucker-Lewis Index (TLI)	≥ 0.95 indicates good fit (Hu and Bentler, 1999; Schumacker and Lomax, 2016)
Standardized Root Mean-square Residual (SRMR)	< 0.05 indicates close fit (Schumacker and Lomax, 2016); $<$ 0.08 indicates good fit (Hu and Bentler, 1999)
χ^2 / degrees of freedom (df)	Ratio of < 2 indicates good fit (Tabachnick and Fidell, 2013)

[1] This test fails for many models and is sensitive to sample size. Many SEM researchers agree that failure of this test does not need to lead to rejection of a model. We report it here for completeness.

The recommended thresholds and cut-off values are rules of thumb and should be used with care. There is considerable discussion among SEM scholars on the validity of these fit measures, and on which of these are most important. Nevertheless, the fit measures reported here are most commonly reported. It is custom to assess a model by looking at several of these fit measures, and judge a model based on the scores of the majority of these fit measures. In that sense, the various measures of fit can be interpreted as 'indicators' of the latent construct 'model fit' (cf. Fig. 4).[4] We accept that each measure of fit has its weaknesses, but each takes a different approach to measure (mis)fit. Taken together they can offer some level of confidence in the model, in particular when they all fall within commonly accepted ranges.

One assumption of using a ML estimator is that the variables are continuous, which Likert scales are not by definition. However, the use of ML is unproblematic when applied on Likert-scale variables with a seven-point scale (Rhemtulla et al., 2012; Tarka, 2017); in this study we used a seven-point Likert scale for most of the self-reported data.

4.6 Evaluation of the Measurement Model

We evaluated the measurement model with a confirmatory factor analysis (CFA), whereby all constructs are correlated to one another. The developer engagement construct was modeled as a second-level construct, whereby the constituent constructs ID and HC are modeled as "indicators," and not correlated to other constructs in the CFA. The standard-

[4] Kenneth Bollen, a SEM expert, made this observation verbally several years ago; confirmed in personal communication July 2021.

ized loadings of ID and HC onto developer engagement were above the recommended minimum of 0.6 (Schumacker and Lomax, 2016), namely 0.91 and 0.75, respectively.

The CFA results exhibited a very good fit (p-value χ^2 = .06, RMSEA = 0.04 (CI 0.01–0.06), p-value RMSEA < .05 = 0.74, CFI = 0.98, TLI = 0.97, SRMR = 0.05, χ^2 = 98.4, df = 78, and χ^2/df = 1.3).

Table 4 Confirmatory factor analysis: factor loadings. DE1 and DE2 refer to ID and HC, respectively, as items of the second order construct Developer Engagement. HC=Human Capital, JS=Job Satisfaction, GP=Gamification Participation, ID=Identification with Community, DL=Desire to Learn, EX=Developer Expertise, DE=Developer Engagement

Item	HC	JS	GP	ID	DL	EX	DE
HC1	0.91						
HC2	0.92						
HC3	0.93						
JS1		0.80					
JS2		0.71					
JS4		0.76					
GP1			0.72				
GP2			0.71				
GP3			0.85				
ID1				0.84			
ID2				0.93			
DL1					0.93		
DL2					0.79		
EX1						0.68	
EX2						0.92	
DE1 (ID)							0.90
DE2 (HC)							0.76

4.6.1 Convergent Validity

Convergent validity refers to how well the indicators of a given construct belong together—i.e., how well they converge in their measurement of the construct. A common measure to assess convergent validity is the Average Variance Extracted (AVE), which should be at least 0.5 (Fornell and Larcker, 1981). As Table 5 indicates, the lowest AVE value in our study was 0.58. A second approach to assess convergent validity is to inspect the loadings of the items onto their respective constructs (see Table 4). A common rule of thumb is that standardized loadings be at least 0.5, or preferably 0.7 (Hair, Jr. et al., 2013, p. 617); the lowest loading in our model was 0.68. Other loadings varied from 0.71 to 0.93. We conclude that there were no threats to convergent validity.

4.6.2 Discriminant Validity

Discriminant validity refers to the extent to which the various constructs measure distinct concepts. We assessed this using two procedures.

First, we assessed the Fornell-Larcker criterion (Fornell and Larcker, 1981), which requires that all square root values of the AVEs exceed the correlations between the

Table 5 Average Variance Extracted (AVE) and correlations among the constructs and square roots of the AVE values (in boldface on diagonal)

Construct	AVE	1	2	3	4	5
1. Desire to learn	0.72	**0.85**				
2. Expertise	0.58	0.64	**0.76**			
3. Participation	0.72	0.76	0.41	**0.85**		
4. Developer engagement	0.69	0.31	0.28	0.45	**0.83**	
5. Job satisfaction	0.58	0.08	0.06	0.00	0.25	**0.76**

Table 6 Heterotrait Monotrait (HTMT) ratios of first-order constructs

Construct	1	2	3	4	5
1. Desire to learn					
2. Expertise	0.64				
3. Participation	0.73	0.46			
4. Identification	0.30	0.29	0.44		
5. Human capital	0.18	0.18	0.32	0.69	
6. Job satisfaction	0.09	0.09	0.07	0.17	0.27

constructs. Table 5 confirms that this was the case.[5] Second, we considered the Hetero-Trait Mono-Trait (HTMT) ratios (Henseler et al., 2015) (see Table 6). The HTMT ratios are calculated only for the first-order constructs, and not for the second-order construct (Sarstedt et al., 2019, p. 203). A liberal cut-off value of the HTMT ratios is 0.9, beyond which discriminant validity could be problematic; a more conservative cut-off value is 0.85 (Henseler et al., 2015). The highest HTMT ratio in our study was 0.73, well below the conservative cut-off. Based on these results, we conclude that there were no threats to discriminant validity.

4.7 Common Method Variance

Common method variance (CMV) (also referred to as common method bias) refers to variance that is attributable to the measurement method (Podsakoff et al., 2003). That is, the mere use of the same method to collect data representing predictor and criterion variables can lead to artifactual covariance between these variables; Podsakoff et al. discuss a wide range of potential sources of CMV (Podsakoff et al., 2003). We have included a procedural remedy to control for CMV, by drawing on two separate data sources: Desire to Learn, Expertise, and Gamification Participation were measured using data from the company's private GitLab installation, whereas all other constructs were measured using the survey instrument. Using separate data sources is the "gold standard" to address CMV (Jordan and Troth, 2020). Nevertheless, given that our theoretical model includes a mediator, which serves as both an exogenous and endogenous variable, we also conducted Harman's single-factor test. Using this procedure, all variables in the study (from the same source) are loaded onto a single factor in an exploratory factor analysis, relying on an unrotated factor solution (Podsakoff et al., 2003, p. 889). CMV may be present if a single factor emerges, or the first factor accounts for the majority of

[5] To be precise, the square root of the AVE of Developer Expertise was 0.764, and the correlation between Participation and Desire to learn was 0.762; both rounded down to 0.76.

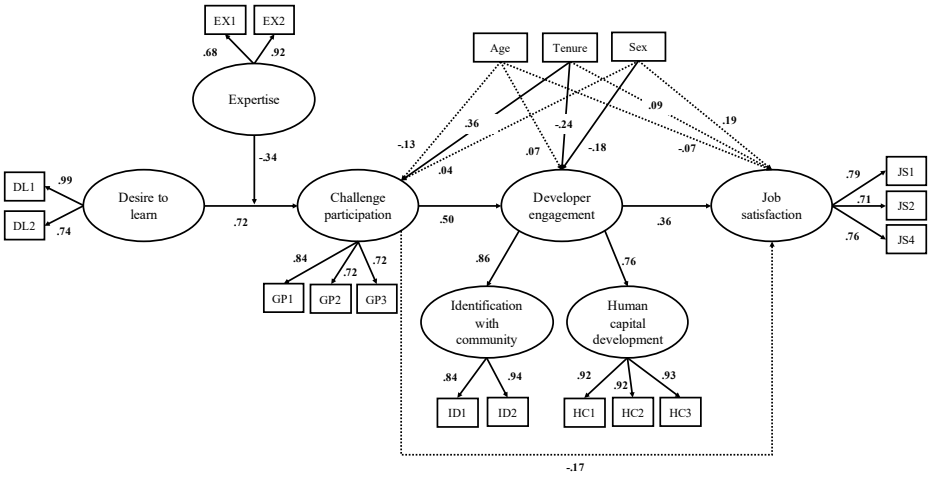

Fig. 5. Results of the structural equation model. Dotted lines indicate non-significant results. All estimates are standardized. Loadings of Expertise taken from the CFA. Moderation analysis performed using factor scores in a separate model.

the variance. The first factor in our results accounted for 29.3%, which is well below the cut-off of 50%. This, plus the procedural strategy of collecting data from two different data sources suggest that CMV was not a concern in this study.

5 Results and Discussion

We now present the results of the evaluation of the theoretical model (Sec. 5.1), followed by a discussion of implications of these findings (Sec. 5.2). Sec. 5.3 discusses the limitations of this study. Sec. 5.4 concludes with a brief summary.

5.1 Evaluation of the Theoretical Model

This section presents the results of the evaluation of the theoretical model in Fig. 1. Fig. 5 presents the analysis results. Based on commonly used fit measures for evaluating structural equation models (Table 3), the model exhibited a very good fit (p-value χ^2 = 0.02,[6] RMSEA = 0.047 (CI 0.021–0.067), p-value RMSEA $< .05$ = 0.58, CFI = 0.97, TLI = 0.97, SRMR = 0.06, χ^2 = 119.3, df = 89, and χ^2/df = 1.3), which means that overall, the data support the theoretical model. As emphasized in Sec. 4.5, a good model fit is a prerequisite in order to be able to interpret the estimated parameters of the model; a poor model fit essentially renders the model unusable because the parameter estimates cannot be trusted. Table 7 reports the results of the evaluation, including the unstandardized coefficients (B), standardized coefficients (β), standard errors, confidence intervals and p-values.

Hypothesis 1 suggested a positive relationship between a developer's desire to learn and participation in the gamification platform; we found strong support for this (β = 0.72,

[6] As discussed above, a significant χ^2 test is not considered an issue.

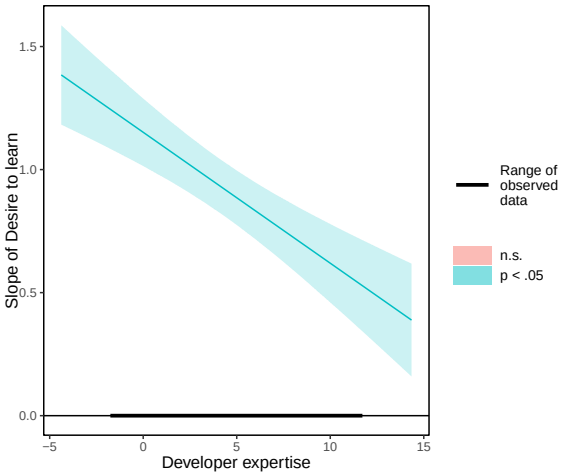

Fig. 6. Johnson-Neyman plot of developer expertise as a moderator of the slope of Desire to learn. Developers who have more skills and know more languages will have a weaker relationship between desire to learn and participation.

$p < 0.001$). Of the three control variables, only tenure was significant ($\beta = 0.36$, $p < 0.01$), indicating that developers with a longer tenure at the company have a higher level of participation.

Hypothesis 2 proposed a negative moderating role for a developer's level of expertise on the relationship between desire to learn and participation; our analysis demonstrated that the data support this moderating relationship ($\beta = -0.34$, $p = 0.00$). Figure 6 shows the Johnson-Neyman plot, which indicates that as developer expertise increases, the slope of Desire to learn decreases. In other words, the relationship between Desire to learn (the predictor) and participation (the dependent variable) becomes weaker as developer expertise is higher. That is, developers with a high level of expertise exhibit a lower level of participation when their desire to learn is high. Developers with a low level of expertise exhibit a higher level of participation with the same high desire to learn. Thus, a higher level of expertise negatively moderates the level of participation. Similar to H1, only tenure was found to be significant ($\beta = 0.16$, $p < 0.01$).

Hypothesis 3 proposed a positive relationship between participation and developer engagement. Our analysis revealed the data support this ($\beta = 0.50$, $p < 0.001$). Sex was found to be significant ($\beta = -0.18$, $p = 0.04$); the negative sign indicates that lower values for sex (i.e. 0, indicating females) link to a higher level of engagement. In other words, females exhibited a slightly higher level of engagement. Tenure was also significant ($\beta = -0.24$, $p = 0.024$); again, the negative sign indicates that developers with shorter tenure exhibited a higher level of engagement.

Furthermore, Hypothesis 4 proposed a positive relationship between developer engagement and job satisfaction. We found support for this hypothesis as well ($\beta = 0.36$, $p = 0.02$). None of the control variables were significant, but sex was borderline insignificant ($\beta = 0.19$, $p = .051$).

Hypothesis 5, finally, proposed a mediating role for developer engagement between participation and job satisfaction. To evaluate a mediation effect, we must consider both

the direct and indirect paths from participation to the outcome variable, job satisfaction (Zhao et al., 2010). We first consider the direct path; the direct path from participation to job satisfaction is not significant ($p = 0.14$). Next, we consider the indirect path. The analysis results indicate that the indirect path from participation to job satisfaction, mediated by developer engagement, is statistically significant ($\beta = 0.18$, $p = 0.02$). As recommended in the SEM literature (Hayes, 2018), we also evaluated the indirect effect with a bootstrapping procedure with 2000 samples. This is because the parameter estimate of an indirect effect is nothing else than a product term, but the distribution of that product term does not follow the same distribution as the original product factors. We refer interested readers to Hayes (2018)'s treatment on this topic. The results were similar and consistent: $\beta = 0.18$, $p = 0.04$.

The result of a mediation analysis can result in no mediation effect, a partial mediation effect, or a full mediation effect (Zhao et al., 2010). Given that there is no significant direct effect but a significant indirect effect only, we conclude that developer engagement fully mediates the relationship between participation and job satisfaction on the other.[7]

Table 7 Unstandardized (B) and standardized (β) path coefficients, standard errors (SE), confidence intervals (CI), and p-values

Hypothesis	B	β	SE	95% CI	p
H1: Desire to learn → Gamification participation	3.99	0.72	0.57	(2.85, 5.15)	0.00
Sex	1.45	0.04	2.25	(−2.97, 5.86)	0.52
Age	−0.25	−0.13	0.14	(−0.52, 0.03)	0.08
Tenure	0.91	0.36	0.25	(0.41, 1.40)	0.00
H2: Expertise negatively moderates Desire to learn → Gamification participation	−0.05	−0.34	0.01	(−0.03, −0.05)	0.00
Sex	−0.01	0.00	0.22	(−0.43, 0.41)	0.97
Age	−0.02	−0.06	0.01	(−0.04, 0.01)	0.17
Tenure	0.06	0.16	0.02	(0.02, 0.11)	<0.01
H3: Gamification participation → Developer engagement	0.04	0.50	0.01	(0.02, 0.05)	0.00
Sex	−0.45	−0.18	0.22	(−0.88, −0.02)	0.04
Age	0.01	0.07	0.01	(−0.02, 0.04)	0.48
Tenure	−0.04	−0.24	0.02	(−0.08, −0.01)	0.02
H4: Developer engagement → Job satisfaction	0.40	0.36	0.17	(0.07, 0.73)	0.02
Sex	0.52	0.19	0.27	(0.00, 1.04)	>0.05
Age	−0.01	−0.07	0.02	(−0.04, 0.02)	0.52
Tenure	0.02	0.09	0.02	(−0.02, 0.06)	0.32
Mediation Analysis					
H5 Direct effects: Participation → Job satisfaction	−0.01	−0.17	0.01	(−0.03, 0.00)	0.09
H5 Indirect effects: Participation → Developer engagement → Job satisfaction	0.01	0.18	0.01	(0.00, 0.03)	0.02

[7] If both the direct effect and indirect effect are statistically significant, this is characterized as a partial mediation. In this case, there is no direct effect, so we speak of 'full' mediation.

Table 8 Variance Explained

Construct	R^2
Gamification Participation	0.78
Developer Engagement	0.26
Job Satisfaction	0.11

Table 8 lists the R^2 values for each of the endogenous variables. The table shows that the variance explained for job satisfaction is limited at 0.11. The values for gamification participation and developer engagement are considerably higher, at 0.78 and 0.26, respectively. While the R^2 values are somewhat informative, we emphasize that these do not signify model fit in the same way they are interpreted in a multiple regression. Clearly, other constructs are at play in explaining job satisfaction, which are not included in our theoretical model because these were not the focus of our study.

5.2 Discussion and Implications

The SE literature features few empirical studies of gamification in industry settings (see Table 1), and in light of this, the present study makes a number of contributions.

First, we conducted a study of gamification in a large multi-site software development organization. We describe the dedicated gamification platform designed around 'exotic' cartoon-style monsters who inhabit an environment consisting of several islands (see Fig. 2), and the 'challenge' mechanism to attract developers. A playful interface helped to distinguish the system from other corporate systems that exhibited a more dull appearance. The system was designed to entice developers to get involved and build a community, and additional, offline activities such as technology meetups offered tangible benefits to community members. Our study is different from prior quantitative studies of gamification in SE in that it offers a larger theoretical framework. We observed that many prior studies reported inconclusive results. One potential explanation for this is that the theorized hypotheses in those studies overlooked more complex relationships between variables. That is, relationships between variables may be more nuanced; mediating and moderating factors may play a role. Without modeling these, the focus remains on direct, linear relationships only. As our study bears out, the link between gamification participation and job satisfaction is mediated by the construct developer engagement. Thus, we suggest that future quantitative work adopts a more theoretically holistic approach that explores "how" questions: how do two (or more) variables relate to one another, and what might underlying mechanisms that link these variables be?

Second, we observed that the term "developer engagement" had not been defined in the software engineering literature, or beyond, and proposed a definition that is grounded in, and consistent with prior literature, and is also measurable through an empirical measurement instrument using latent variables. Other studies can use this definition, either in its current form ('as-is') or as a foundation to extend or adapt; alternatively, we envisage that other scholars may wish to conceptualize the construct of developer engagement in a different fashion. We do not argue that our definition represents the 'final word' on this; in fact, we encourage and welcome others to offer alternatives and enrich our understanding of this important concept that has hitherto been used, but not defined.

Third, we drew on Personal Investment theory and study a salient reason for why software developers might participate in a gamification initiative, namely, a desire to learn new software development skills and technologies (see Beecham et al. (2008) for example). To the best of our knowledge, this motivation has not been linked to participation in gamification initiatives in SE (see Table 1); prior studies have focused primarily on whether gamification works to achieve better quality documentation, code, and so on. Much of the gamification literature (beyond the software engineering literature) refers to self-determination theory in seeking why participants participate, and so the use of an alternative motivational lens contributes to the heterogeneity of theoretical perspectives that Rapp et al. (2019) have called for. Our findings offer support for the hypothesis that developers are interested to participate in a gamification platform if this offers them opportunities to learn new skills and technologies. Organizations that wish to introduce new tools or technologies could consider using a dedicated gamification platform to introduce new tools and technologies, for example, to entice developers to learn a new programming language for use in future projects.

Fourth, we identified a key outcome in the SE literature that is important for software organizations in general, job satisfaction, and found support for our hypotheses that (a) participation is positively associated with developer engagement, (b) developer engagement positively associates with job satisfaction, and (c) that developer engagement mediates the relationship between participation in a gamification platform and job satisfaction. Further, we found that there is no link between participation and job satisfaction directly; in other words, participation in gamification challenges does not explain job satisfaction. Rather, participation induces developer engagement which in turn links to job satisfaction, thus identifying developer engagement as a theoretical explanation of how participation "translates" into job satisfaction. However, we note that the effect size of the mediation is moderate ($\beta = .18$) and a low level of explained variance ($R^2 = .11$) of the job satisfaction construct. This indicates that developer engagement only plays a limited role in explaining developer job satisfaction. Clearly other factors are at play, and future work could focus on other factors, besides developer engagement, to explain developer job satisfaction. While there is considerable qualitative work in this area, we observe a lack of quantitative study designs that seek to explain job satisfaction.

In addition to the substantive contributions to gamification in a software engineering context, this article also makes a novel methodological contribution. This article reports a study using covariance-based structural equation modeling (as opposed to PLS-SEM which is more commonly used in the SE literature (Russo and Stol, 2021)). CB-SEM is an analysis approach that is not widely used in the mainstream software engineering publications (with only a few exceptions (Lindsjørn et al., 2016; Capra et al., 2008; Ralph et al., 2020)). Our study relies on multiple data sets, combining data from a developer survey with data recorded in the company's version control system. Some of the variables collected from the version control system represents developer behavior, whereas the developer survey collected developer opinions. Combining different data sets is the primary strategy to prevent CMV (Podsakoff et al., 2003) (see Sec. 4.7). Further, this study is the advanced analyses involving a second-level latent variable, mediation analysis, and moderation analysis. SEM is an important family of techniques in the emergent behavioral software engineering literature as it relies on psychometric measures (Lenberg et al., 2015). As mentioned, SEM is not widely used in an SE context, and to the best of our knowledge, no studies in the SE literature include second-level constructs, mediation, or moderation analysis. In that context, this study and the

Table 9 Substantive and methodological contributions and implications

Limitations of prior literature on gamification in SE	Contributions	Lessons and Implications
Few empirical studies of gamification of software engineering in industry, and many quantitative studies are inconclusive regarding outcomes.	This study adds to the limited gamification in SE literature, describing a bespoke, attractive, and playful gamification platform through which tasks and challenges can be completed. The findings support our theoretical model.	Other organizations can model a gamification platform inspired by the one presented. Integration with the version control system is important to ensure competitions and tasks are relevant. Offline activities such as technology meetups offers additional tangible benefits to community members. Further studies can theorize alternative mediating and moderating factors; the lack of these may explain inconclusive results in several prior studies.
No explicit definition of "developer engagement" offered that is not conflated with participation.	This study proposes an explicit definition grounded in and consistent with prior literature.	The proposed definition can be used as a foundation for future research, as-is, as a starting point to extend, or to offer alternative viewpoints.
Lack of understanding of what drives software developers to partake in a gamification platform (beyond motivational affordances).	This study finds a positive link between developers' desire to learn new skills and technology and participation in gamification challenges. This link is negatively moderated by their level of expertise.	Gamification of specific tasks can help to attract software developers; learning new skills and technologies is an attractor to get developers interested. Organizations could consider the use of a dedicated gamification platform to introduce new technologies.
Little research on the benefits of increasing developer engagement for organizations.	This study demonstrates a positive link between gamification participation and job satisfaction, mediated by developer engagement; however, low level of explained variance (R^2=.11) and moderate effect size (β=.18)	Developer engagement is one 'mechanism' through which developer job satisfaction can rise. However, offering gamification challenges should not be seen as a means to increase developer job satisfaction.
No prior studies of gamification in SE used covariance-based structural equation modeling	This study demonstrates the use of CB-SEM on a non-trivial data set from two separate sources, including mediation and moderation hypotheses, and the use of a second-level latent variable.	This paper offers a starting point for SE researchers who seek to measure and analyze latent variables and develop new theory using SEM.

replication package with the necessary scripts and instructions to conduct these analysis may be welcomed by researchers seeking to contribute to the behavioral SE literature (Lenberg et al., 2015).

5.3 Limitations of this Study

This study has a number of limitations. This study design is not experimental but is a sample study; therefore, strictly speaking we cannot claim any causality. We cannot with full certainty establish that, for example, gamification participation *causes* a higher level of developer engagement. However, the logic of SEM studies is that the tested theory is

firmly grounded in prior literature. The purpose of SEM studies is not to establish causal links, but to *confirm* a theory from prior literature and propose associations between key concepts. Before any of the estimated parameters are considered, in SEM studies such as this it is critical that the 'fit' of the model as a whole is considered. Model fit indicators (see Table 3) assess the degree to which the 'structure' of the data matches the structure implied by the theoretical model. In this study, we have grounded the hypotheses firmly in prior literature, and so the justification of these hypotheses is a convergent result of that literature review. While one could potentially link any two concepts and attempt to establish a statistical significant result, without a firm theoretical foundation, such findings would bear no value, keeping in mind that analysis results must also have face validity (Bollen, 2011). It is in this light that the value of this study should be evaluated: it presents evidence in support of the theoretical model that was carefully composed based on prior studies and theories. Further, a single study should not be interpreted as the final word on the topic; it provides a single data point for future meta-analyses and literature review syntheses which are more suitable to detect patterns of results in which we can have more confidence.

The data for this study were collected at a single organization. This was necessary because some of the constructs had unique measurement instruments that would not be readily applicable to any other organization. The data were thus collected within a specific firm setting, and within the context of an inner source initiative (Cooper and Stol, 2018). Hence, the findings of this study are not statistically generalizable to other organizational settings. However, the findings of this study demonstrate that the proposed theory is indeed applicable to this one organization, and further, the theoretical model could be applied to other organizations with similar gamification platforms. Measurement of some constructs might have to be tailored to other settings. Summarizing, while statistical generalization is limited, findings of this study can be seen to contribute to theoretical generalizability.

5.4 Conclusion

Gamification has started to receive increasing attention in the software engineering literature. Many publications, however, seem to report non-empirical studies that propose visions of gamified software development (Pedreira et al., 2015; García et al., 2017), and much of the empirical work is conducted with students, rather than in industry settings, with some notable exceptions (Herranz et al., 2018; Neto et al., 2018; Marques et al., 2020; Snipes et al., 2014; Passos et al., 2011). Further, many studies in the software engineering domain thus far have reported inconclusive results. We posited that this might be the case because prior studies have not considered more complex types of relationships between variables involving mediating and moderating mechanisms. Gamification has been proposed to increase developer engagement, but no definition of the term has been proposed. Further, while developer engagement sounds like an appealing value proposition for software organizations, there are no studies that focus on the consequences of having engaged developers. Hence, in this study we sought to address these issues. This paper contributes to the limited body of knowledge on gamification in software engineering by reporting a rigorous empirical study of gamification in one software development organization. We proposed a theoretical model that, amongst others, posits that developers' desire to learn new skills and technologies positively

links to participation in gamification challenges, but that this relationship is moderated by developer expertise. Further, we posit that the relationship between participation in gamification challenges and our dependent variable of interest, job satisfaction, is mediated by the construct developer engagement, for which we offer a definition based on prior literature—hitherto the term engagement had been conflated with 'participation.' We argue that participation leads to a higher level of engagement, and that this in turn translates into job satisfaction. Thus, we demonstrate that relationships between variables of interest may not be straightforward, and that these require more thoughtful theorizing. This should be a consideration for future studies of gamification in the software engineering domain. Structural equation modeling as a relatively novel approach within the SE domain is an appropriate technique to perform such analyses. Using combined data from two data sources at an organization that had started a gamification platform, we found that the data support our theoretical model. Based on our findings, we discussed the potential that gamification platforms might have for other software organizations. We hope that this study represents a starting point for more industry studies of gamification in the software sector.

References

Adams PJ, Capiluppi A, de Groot A (2008) Detecting agility of open source projects through developer engagement. In: IFIP International Conference on Open Source Systems, Springer, pp 333–341

Alhammad MM, Moreno AM (2018) Gamification in software engineering education: A systematic mapping. J Syst Software 141:131–150

Alhammad MM, Moreno AM (2020) Challenges of gamification in software process improvement. Journal of Software: Evolution and Process 32(6):e2231

Aurum A, Daneshgar F, Ward J (2008) Investigating knowledge management practices in software development organisations: an australian experience. Inf Softw Technol 50:511–533

Badihi S, Heydarnoori A (2017) Crowdsummarizer: Automated generation of code summaries for java programs through crowdsourcing. IEEE Software 34(2):71–80

Bagozzi R, Dholakia U (2006) Open source software user communities: a study of participation in Linux user groups. Manage Sci 52:1099–1115

Beecham S, Baddoo N, Hall T, Robinson H, Sharp H (2008) Motivation in software engineering: A systematic literature review. Inf Softw Technol 50(9-10):860–878

Bitrián P, Buil I, Catalán S (2021) Enhancing user engagement: The role of gamification in mobile apps. Journal of Business Research 132:170–185

Bollen KA (2011) Evaluating effect, composite, and causal indicators in structural equation models. MIS Quarterly 35(2):359–372

Braskamp L (2009) Applying Personal Investment Theory to Better Understand Student Development, Information Age Publishing

Capra E, Francalanci C, Merlo F (2008) An empirical study on the relationship among software design quality, development effort, and governance in open source projects. IEEE Trans Softw Eng 34(6):765–782

Cheong C, Cheong F, Filippou J (2013) Quick quiz: A gamified approach for enhancing learning. In: Pacific-Asia Conference on Information Systems

Chiu CM, Hsu MH, Wang ET (2006) Understanding knowledge sharing in virtual communities: An integration of social capital and social cognitive theories. Decision Support Systems 42(3):1872–1888

Cooper D, Stol KJ (2018) Adopting InnerSource: Principles and Case Studies. O'Reilly Media

Dabbish L, Stuart C, Tsay J, Herbsleb J (2012) Social coding in GitHub: transparency and collaboration in an open software repository. In: Conference on Computer Supported Cooperative Work, ACM, pp 1277–1286

Dal Sasso T, Mocci A, Lanza M, Mastrodicasa E (2017) How to gamify software engineering. In: 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, pp 261–271

de Melo AA, Hinz M, Scheibel G, Diacui Medeiros Berkenbrock C, Gasparini I, Baldo F (2014) Version control system gamification: A proposal to encourage the engagement of developers to collaborate in software projects. In: Meiselwitz G (ed) Social Computing and Social Media, Springer, pp 550–558

Deterding S (2012) Gamification: designing for motivation. Interactions 19(4):14–17

Deterding S, Dixon D, Khaled R, Nacke L (2011) From game design elements to gamefulness: defining "gamification". In: 15th International Academic MindTrek Conference: Envisioning Future Media Environments, pp 9–15

Devi VR (2009) Employee engagement is a two-way street. Human Resource Management International Digest 17(2):3–4

Dong T, Dontcheva M, Joseph D, Karahalios K, Newman M, Ackerman M (2012) Discovery-based games for learning software. In: 2012 ACM annual conference on Human Factors in Computing Systems, ACM, pp 2083–2086

Dubois DJ, Tamburrelli G (2013) Understanding gamification mechanisms for software development. In: 9th Joint Meeting on Foundations of Software Engineering, ACM, p 659–662

Finney S, DiStefano C (2012) Non-normal and categorical data in structural equation modeling. In: Hancock G, Mueller R (eds) Structural Equation Modeling: A Second Course, 2nd edn, Information Age Publishing

Fornell C, Larcker D (1981) Evaluating structural equation models with unobservable variables and measurement error. Journal of Marketing Research 18(1):39–50

França C, da Silva F, Sharp H (2020) Motivation and satisfaction of software engineers. IEEE Transactions on Software Engineering 46(2):118–140

Fraser G (2017) Gamification of software testing. In: 2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST), IEEE, pp 2–7

Furr M, Bacharach V (2008) Psychometrics: An Introduction. Sage

Gana K, Broc G (2019) Structural Equation Modeling with lavaan. Wiley

García-Mireles GA, Morales-Trujillo ME (2019) Gamification in software engineering: A tertiary study. In: International Conference on Software Process Improvement, Springer, pp 116–128

García F, Pedreira O, Piattini M, Cerdeira-Pena A, Penabad M (2017) A framework for gamification in software engineering. J Syst Software 132:21–40

Grant S, Betts B (2013) Encouraging user behavior with achievements: an empirical study. In: 10th Working Conference on Mining Software Repositories, pp 65–68

Graziotin D, Lenberg P, Feldt R, Wagner S (2020) Behavioral software engineering: Methodological introduction to psychometrics. arXiv preprint arXiv:200509959

Hair JF, Hult GTM, Ringle CM, Sarstedt M (2016) A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM)

Hair, Jr J, Black WC, Babin BJ, Anderson RE (2013) Multivariate Data Analysis, 7th edn. Pearson Education Ltd.

Hamari J, Koivisto J, Sarsa H (2014) Does gamification work? a literature review of empirical studies on gamification. In: 47th Hawaii Int'l Conf. System Sciences, pp 3025–3034

Hayes A (2018) Introduction to mediation, moderation, and conditional process analysis: A regression-based approach. The Guilford Press

Henseler J, Ringle CM, Sarstedt M (2015) A new criterion for assessing discriminant validity in variance-based structural equation modeling. Journal of the Academy of Marketing Science 43(1):115–135

Herranz E, Colomo-Palacios R, de Amescua Seco A (2015) Gamiware: a gamification platform for software process improvement. In: European Conference on Software Process Improvement, Springer, pp 127–139

Herranz E, Guzmán JG, de Amescua-Seco A, Larrucea X (2018) Gamification for software process improvement: a practical approach. IET Softw 13(2):112–121

Hertel G, Niedner S, Herrmann S (2003) Motivation of software developers in open source projects: an internet-based survey of contributors to the Linux kernel. Res Policy 32:1159–1177

Hu L, Bentler P (1999) Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. Struct Equ Modeling 6(1):1–55

Jordan P, Troth A (2020) Common method bias in applied settings: The dilemma of researching in organizations. Australian Journal of Management 45(1):3–14

Kahn W (1990) Psychological conditions of personal engagement and disengagement at work. Academy of Management Journal 33:692–724

Karanika-Murray M, Duncan N, Pontes HM, Griffiths MD (2015) Organizational identification, work engagement, and job satisfaction. J Manag Psychol 30(8):1019–1033

Khandelwal S, Sripada SK, Reddy YR (2017) Impact of gamification on code review process: An experimental study. In: Proceedings of the 10th Innovations in Software Engineering Conference, pp 122–126

Kline R (2016) Principles and Practice of Structural Equation Modeling, 4th edn. Guilford Press

Kline RB (2012) Assumptions in structural equation modeling. In: Hoyle RH (ed) Handbook of structural equation modeling, The Guilford Press

Lenberg P, Feldt R, Wallgren LG (2015) Behavioral software engineering: A definition and systematic literature review. J Syst Software 107:15–37

Li W, Grossman T, Fitzmaurice G (2012a) Gamicad: a gamified tutorial system for first time autocad users. In: 25th Symposium on User Interface Software and Technology, pp 103–112

Li Z, Huang KW, Cavusoglu H (2012b) Quantifying the impact of badges on user engagement in online Q&A communities. In: Thirty-third International Conference on Information Systems

Lindsjørn Y, Sjøberg DI, Dingsøyr T, Bergersen GR, Dybå T (2016) Teamwork quality and project success in software development: A survey of agile development teams. J Syst Software 122:274–286

Loehlin JC, Beaujean AA (2017) Latent Variable Models: An Introduction to Factor, Path, and Structural Equation Analysis, 5th edn. Routledge

Lombriser P, Dalpiaz F, Lucassen G, Brinkkemper S (2016) Gamified requirements engineering: Model and experimentation. In: Requirements Engineering: Foundation for Software Quality, Springer International Publishing, pp 171–187

Longo M, Mura M (2011) The effect of intellectual capital on employees' satisfaction and retention. Information & Management 48:278–287

Macey W, Schneider B (2008) The meaning of employee engagement

Machuca-Villegas L, Gasca-Hurtado GP (2018) Gamification for improving software project management processes: a systematic literature review. In: International Conference on Software Process Improvement, Springer, pp 41–54

Maehr M, Braskamp L (1986) The motivation factor: A theory of persona investment. Heath & Co.

Mäntylä MV, Smolander K (2016) Gamification of software testing-an mlr. In: International Conference on Product-Focused Software Process Improvement, Springer, pp 611–614

Marczewski A (2013) Gamification: A Simple Introduction. Lulu Press

Marques R, Costa G, Mira da Silva M, Goncalves D, Goncalves P (2020) A gamification solution for improving scrum adoption. Empir Software Eng 25:2583–2629

Meldrum S, Licorish SA, Savarimuthu BTR (2017) Crowdsourced knowledge on Stack Overflow: A systematic mapping study. In: 21st International Conference on Evaluation and Assessment in Software Engineering, ACM, p 180–185

Morschheuser B, Hamari J, Koivisto J (2016) Gamification in crowdsourcing: A review. In: 2016 49th Hawaii International Conference on System Sciences (HICSS), pp 4375–4384

Morschheuser B, Hassan L, Werder K, Hamari J (2018) How to design gamification? a method for engineering gamified software. Information and Software Technology 95:219 – 237

Neto PS, Medeiros DB, Ibiapina I, da Costa Castro OC (2018) Case study of the introduction of game design techniques in software development. IET Softw 13(2):129–143

Passos EB, Medeiros DB, Neto PAS, Clua EWG (2011) Turning real-world software development into a game. In: Brazilian Symposium on Games and Digital Entertainment, pp 260–269

Pedreira O, García F, Brisaboa N, Piattini M (2015) Gamification in software engineering – a systematic mapping. Information and Software Technology 57:157–168

Podsakoff P, MacKenzie S, Jeong-Yeon L, Podsakoff N (2003) Common method biases in behavioral research: A critical review of the literature and recommended remedies. Journal of Applied Psychology 88(5):879–893

Prause CR, Nonnen J, Vinkovits M (2012) A field experiment on gamification of code quality in agile development. In: roceedings of the 24th Annual Workshop of the Psychology of Programming Interest Group (PPIG'2012), p 17

Ralph P, Baltes S, Adisaputri G, Torkar R, Kovalenko V, Kalinowski M, Novielli N, Yoo S, Devroey X, Tan X, Zhou M, Turhan B, Hoda R, Hata H, Robles G, Fard AM, Alkadhi R (2020) Pandemic programming: How COVID-19 affects software developers and how their organizations can help. Empirical Software Engineering

Rapp A, Hopfgartner F, Hamari J, Linehan C, Cena F (2019) Strengthening gamification studies: Current trends and future opportunities of gamification research. International Journal of Human-Computer Studies 127:1–6

Reinartz W, Haenlein M, Henseler J (2009) An empirical comparison of the efficacy of covariance-based and variance-based SEM. International Journal of Research in Marketing 26(4):332–344

Rhemtulla M, Brosseau-Liard P, Savalei V (2012) When can categorical variables be treated as continuous? a comparison of robust continuous and categorical SEM estimation methods under suboptimal conditions. Psychol Methods 17(3):354–373

Richter G, Raban DR, Rafaeli S (2015) Studying gamification: The effect of rewards and incentives on motivation. In: Reiners T, Wood LC (eds) Gamification in Education and Business, Springer International Publishing, Cham, pp 21–46

Rigdon EE, Sarstedt M, Ringle CM (2017) On comparing results from CB-SEM and PLS-SEM: Five perspectives and five recommendations. Marketing: ZFP – Journal of Research and Management 39(3):4–16

Rosseel Y (2012) lavaan: An R package for structural equation modeling. Journal of Statistical Software 48(2):1–36

RStudio Team (2020) RStudio: Integrated Development Environment for R. RStudio, Inc., Boston, MA

Russo D, Stol KJ (2021) Pls-sem for software engineering research: An introduction and survey. ACM Computing Surveys 54(4)

Saks AM (2006) Antecedents and consequences of employee engagement. J Manag Psychol 21(7):600–619

Sarstedt M, Hair JF, Cheah JH, Becker JM, Ringle CM (2019) How to specify, estimate, and validate higher-order constructs in PLS-SEM. Australas Mark J 27(3):197–211

Savalei V (2018) On the computation of the RMSEA and CFI from the mean-and-variance corrected test statistic with nonnormal data in sem. Multivariate Behavioral Research 53(3):419–429

Schaarschmidt M, Homscheid D, Kilian T (2019) Application developer engagement in open software platforms: An empirical study of Apple iOS and Google Android developers. International Journal of Innovation Management 23(4)

Schaufeli W (2013) What is engagement? In: Truss C, Alfes K, Delbridge R, Shantz A, Soane E (eds) Employee Engagement in Theory and Practice, Routledge

Schumacker R, Lomax R (2016) A Beginner's Guide to Structural Equation Modeling, 4th edn. Routledge

Sharma GG, Stol KJ (2020) Exploring onboarding success, organizational fit, and turnover intention of software professionals. J Syst Software 159:1–16

Singer L (2013) Improving the adoption of software engineering practices through persuasive interventions. PhD thesis, Gottfried Wilhelm Leibniz Universität Hannover

Singer L, Schneider K (2012) It was a bit of a race: Gamification of version control. In: 2nd International Workshop on Games and Software Engineering (GAS), pp 5–8

Snipes W, Nair AR, Murphy-Hill E (2014) Experiences gamifying developer adoption of practices and tools. In: 36th International Conference on Software Engineering

Stol KJ, Fitzgerald B (2018) The ABC of software engineering research. ACM Transactions on Software Engineering and Methodology 27(3)

Stol KJ, Caglayan B, Fitzgerald B (2017a) Competition-based crowdsourcing software development: A multi-method study from a customer perspective. IEEE Trans Softw Eng 45(3):237–260

Stol KJ, LaToza T, Bird C (2017b) Crowdsourcing for software engineering. IEEE Software 34(2):30–36

Strode D (2016) A dependency taxonomy for agile software development projects. Inf Syst Front 18:23–46

Suff P, Reilly P (2008) Going the Extra Mile: The relationship between reward and employee engagement. Institute for Employment Studies

Sukale R, Pfaff MS (2014) Quodocs: improving developer engagement in software documentation through gamification. In: 32nd ACM Conference on Human Factors in Computing Systems

Tabachnick B, Fidell L (2013) Using Multivariate Statistics, 6th edn. Pearson

Tarka P (2017) The comparison of estimation methods on the parameter estimates and fit indices in sem model under 7-point likert scale. Archives of Data Science 2(1)

Van Doorn J, Lemon KN, Mittal V, Nass S, Pick D, Pirner P, Verhoef PC (2010) Customer engagement behavior: Theoretical foundations and research directions. Journal of Service Research 13(3):253–266

Vargas-Enriquez J, Garcia-Mundo L, Genero M, Piattini M (2015) A systematic mapping study on gamified software quality. In: 7th Int'l Conf. Games and Virtual Worlds for Serious Applications

Vasilescu B, Serebrenik A, Devanbu P, Filkov V (2014) How social Q&A sites are changing knowledge sharing in open source software communities. In: 17th Conference on Computer Supported Cooperative Work & Social Computing, ACM

von Ahn L (2006) Games with a purpose. Comput 39(6):92–94

Welbers K, Konijn EA, Burgers C, de Vaate AB, Eden A, Brugman BC (2019) Gamification as a tool for engaging student learning: A field experiment with a gamified app. E-Learning and Digital Media 16(2):92–109

Williams LJ, O'Boyle Jr E (2011) The myth of global fit indices and alternatives for assessing latent variable relations. Organizational Research Methods 14(2):350–369

Yilmaz M, O'Connor RV (2016) A scrumban integrated gamification approach to guide software process improvement: a Turkish case study. Tehnički vjesnik 23(1):237–245

Yilmaz M, O'Connor RV, Colomo-Palacios R, Clarke P (2019) Guest editorial: Gamification and persuasive games for software engineering. IET Softw 13(2):97–98

Zhao X, Lynch JG, Chen Q (2010) Reconsidering Baron and Kenny: Myths and truths about mediation analysis. Journal of Consumer Research 37(2):197–206

Zichermann G, Cunningham C (2011) Gamification by Design: Implementing game mechanics in web and mobile apps. O'Reilly Media, Inc.