5-2-2023

A DEVOPS PERSPECTIVE: THE IMPACT OF ROLE TRANSITIONS ON SOFTWARE SECURITY CONTINUITY

Hwee-Joo Kam
University of Tampa, hkam@ut.edu

John D'Arcy
University of Delaware, jdarcy@udel.edu

A DEVOPS PERSPECTIVE: THE IMPACT OF ROLE TRANSITIONS ON SOFTWARE SECURITY CONTINUITY

Research in Progress

Hwee-Joo Kam, University of Tampa, United States, hkam@ut.edu

John D'Arcy, University of Delaware, United States, jdarcy@udel.edu

Abstract

Prior studies have claimed that, with evolving technologies, continuity processes in Development and Operations (DevOps) needs to be re-evaluated consistently. In this context, this study examines a relatively new continuity process -- continuous software security. With regulatory compliance and ubiquitous cyberattacks, it becomes increasingly important to integrate security into DevOps. Presently, DevOps developers assume the role of systems operators and software developers to facilitate Continuous Integration/Continuous Deployment (CI/CD). Assuming multiple roles involves role transitioning that requires developers to psychologically disengage from their current role and engage in another. Gradually, this may cause mental exhaustion. Therefore, implementing continuous security may add more complexity to DevOps, instigating more stress to software developers. Given this concern, we draw on Role Transition Theory and Cognitive Load Theory to examine whether integrating security into DevOps would aggravate developers' job burnout, which would eventually undermine continuous security practices.

Keywords: DevOps, information security, software developers, role theory

1 Introduction

The 2022 Verizon Data Breach Investigations Report (DBIR) revealed that exploitation of software application vulnerabilities was responsible for more than 40% of data breaches (Verizon, 2022). To mitigate software vulnerabilities, organizations begin to integrate systems operations and information security (InfoSec) controls into software development (GitLab, 2022). As a result, software developers have to "wear many hats" (GitLab, 2022). In a survey entitled "GitLab 2022 Global DevSecOps Survey", 38% of the surveyed developers reported that, aside from computer programming, they had to monitor their software's operations in a given information technology (IT) platform (GitLab, 2022). Based on this survey, Silverthorne (2022) stated:

> *Today's developers are literally DIYing all the [operations] things. This year, 38% reported instrumenting code they've written for production monitoring, up 12% from 2021 and more than double the percentage in 2020. The same percentage of [developers] monitor and respond to the infrastructure, up 13% from last year…And, in any time left over, [developers] are digging into security, so much so that 53% said they are fully responsible for security in their organizations.*

Asides from developer role, software developers are playing the role of systems operators (GitLab, 2022) that incorporate systems engineering tasks. We argue that systems engineering embodies systems operations, primarily because "*systems engineering is focused on the system as a whole; it emphasizes its total* ***operation****. It looks at the system from the outside, that is, at its interactions with other systems and the environment, as well as from the inside*" (Kossiakoff et al., 2020, p. 4). This suggests that software developers need to play the role of systems operators built on systems engineering. Accordingly, we refer systems operations (i.e., a part of DevOps) to systems engineering tasks that software developers are expected to carry out when playing the systems operator's role.

A role signifies a position with prescribed responsibilities (Ashforth et al., 2000). Role transition involves disengaging from a current role and engaging in another (Richter, 1984). When software developers switch from computer programming to monitoring their software's operations, they have to disengage from programming that embodies software design, syntactic knowledge processing (Bishop-Clark, 1995), and creative problem-solving (Graziotin et al., 2014), in order to adopt systems engineering tasks that require systems configurations through systematic processing (McCumber & Sloan, 2002) and through conceptualization of interrelatedness between various systems components (Kam & Shang, 2019). Such a transition could be mentally exhaustive.

Furthermore, systems operations tasks are often beyond software developers' craft. Software developers were mainly trained to build software rather than to undertake systems complexities in IT systems. Developers may have built distributed, cross-platform software, but such development vastly differs from managing diverse IT systems spanned across different network architectures and boundaries. Overall, developers gain job satisfaction from creative problem-solving (Gallivan, 2003), rather than from managing complexities required in systems operations. This suggests that systems operations tasks might be a poor cognitive fit (i.e., individuals' preference of cognitive style) ascribed to mental distress (Chilton et al., 2005).

We argue that software developers' well-being is critical not only to code productivity and quality (Graziotin et al., 2018), but also to developers' involvement of continuous security practices when security is integrated into DevOps. In general, continuous security practices, which include activities such security testing and planning, promote continuous software security assurance (Kumar & Goyal, 2020). Developers suffering from burnout would not be able to run software security practices properly due to their deteriorated cognitive functions (e.g., decline in cognitive processing) resulted from burnout (Iskander, 2019). When burnout prevents developers to assure software security, software security is undermined. Despite the criticality of this issue, Information Systems (IS) research in this area is scarce. Presently, prior IS studies outline three main themes of information systems development (ISD), namely ISD stakeholders (i.e., human involvement such as users' and managers' participations in ISD), ISD processes (i.e., planned and ad-hoc activities such as pair programming and Kanban), and ISD outputs (i.e., outcomes such as systems reliability and systems maintenance) (Matook et al., 2021). Nevertheless, the criticality of software security has never been mentioned. To fill in the research gap, we investigate the effects of role transitioning on software developers' involvement in continuous software security practices. Our research question is:

> *In the context of information security integration into DevOps, how would role transitions affect software developers' involvement in continuous software security practices?*

Several prior studies examined developers' behaviors using role ambiguity (Rasch & Tosi, 1992; Rezvani & Khosravi, 2019) and role conflict (Venkatesh et al., 2020; Windeler et al., 2017). Role ambiguity refers to the extent of unclear expectations thrust upon software developers, whereas role conflict refers to the extent of inconsistent expectations on how software developers should behave (Rasch & Tosi, 1992; Rizzo et al., 1970). Nevertheless, this study employs Role Transitioning Theory (Ashforth et al., 2000). Role Transitioning Theory presents psychological movements between roles, and posits that micro role transitions entail recurring role transitions that occur with very little spatial and temporal constraints (Ashforth et al., 2000). Particularly, micro role transitions incorporate role segmentation (i.e., distinguished identities between roles) and role integration (i.e., role blurring that allows cross-role interruptions), in which the former infers lesser degree of role ambiguity and role conflict, but the latter suggests otherwise. In DevOps, software developers are often "wearing many hats" (GitLab, 2022) so role transitions occur frequently. With the integration of security, developers need to frequently switch to systems operators and software security tester roles (IBM Cloud Education, 2020). Software developers could possibly experience blurring of role boundaries during micro role transitions (i.e., role integration) because IT working environments usually facilitate software development, software testing, and systems operations. Therefore, we contend that Role Transitioning Theory offers a more viable theoretical framework to examine security behaviors of software developers who are actively participating in DevOps.

Overall, this study contributes to IS research in the following ways. First, we incorporate information security into DevOps in a behavioural study to shed lights on ISD in a DevOps context. Second, this study addresses an up-to-date topic related to software security based on Role Transitioning Theory (Ashforth et al., 2000). The criticality of software security is well recognized in the industry (Edmundson & Hartman, 2022; Verizon, 2022) and academic community (August & Tunca, 2011; Siavvas et al., 2022; Yasin et al., 2019), but the actors (i.e., software developers) who have profound influence on software security have not received as much attention. In this context, our research findings provide insights into one of the factors (i.e., developers' continuous software security practices built on role transitions) that would influence software security.

2 Related Work and Theoretical Background

2.1 Development and Operations

Development and Operations (DevOps) is defined as "a set of continuously improved principles for collaborative work implemented between the IS development function and the IS operations function, along with potentially other stakeholders, which is founded on the sharing of culture, goals, measures, automation tools and automated processes towards continuous delivery of valuable outcome." (Hüttermann, 2021, p. 3). This suggests that DevOps is a process that integrates development and operations based upon the pillars of culture (i.e., addressing cultural barriers), automation (i.e., automated tools for continuous delivery), lean (i.e., eliminating waste and bottleneck), measurement (i.e., shared metrics across functions), and sharing (i.e., shared goals and understanding) (CALMS) (Fitzgerald & Stol, 2017; Hüttermann, 2021).

On the other hand, Maruping & Matook (2020) encouraged DevOps studies to address a tighter integration of orchestration with software, hardware, and community actors (i.e., stakeholders), citing that software development ecosystems have evolved with new technologies such as cloud computing. Built on this notion, Wiedemann et al. (2020) proposed that an integration of development and operations could be achieved through individual componentization (i.e. a flexible IT infrastructure), integrated responsibility (i.e., accountabilities of software deliveries based on collective ownership of software development), and multidisciplinary knowledge (i.e., shared knowledge of software development). Additionally, Hemon-Hildgen et al. (2020) revealed that DevOps developers attained higher job satisfaction in comparison to Agile developers due to better team orchestration.

2.2 Continuous Software Security

To highlight the essence of DevOps, Hemon-Hildgen & Rowe (2022) contended that DevOps was based on continuity, but it was not a methodology defined by a set of rules and procedures. As technologies are perpetually evolving (Maruping & Matook, 2020), it is necessary to re-examine continuity processes in DevOps (Hemon-Hildgen & Rowe, 2022). Therefore, this study addresses a continuity process involving continuous software security when security is integrated into DevOps. Specifically, continuous software security practices encompass continuous testing (i.e., security test that examine software codes and environment), continuous planning, continuous design and development (i.e., security by design and threat modelling), continuous integration (CI)/ continuous deployment (CD) (i.e., automation and configuration management), continuous deployment (i.e. delivery automation), continuous operation (i.e., continuous monitoring and log analysis) and continuous feedback (i.e., team communications) in a security context (Kumar & Goyal, 2020).

2.3 Role Transitioning Theory

With DevOps, developers need to regularly switch between developer role and system operator role. A role represents a position with a set of prescribed responsibilities, and a role boundary defines the scope of a given role (Ashforth et al., 2000). Role transitions represent psychological movements that

involve disengaging from one role and engaging in another (Richter, 1984). In a DevOps context, software developers regularly switch between developers and systems operators (GitLab, 2022). Therefore, role transitions are frequent and do not have to overcome immense physical boundaries such as geographical constraints. As a result, micro role transitions occur (Ashforth et al., 2000).

Micro role transitions are affected by flexibility and permeability of a role boundary (Ashforth et al., 2000). Flexibility refers to the degree of pliability in terms of spatial and temporal boundaries (Hall & Richter, 1988), whereas permeability refers to the degree of role multitasking in which one could physically located at a role's domain and psychologically engaged in another role (Pleck, 1977). In terms of DevOps, we argue that role transitions involve high flexibility and permeability, primarily because, in an IT working environment, software developers can easily perform multi-tasking (e.g., writing and testing codes) and switch to systems operators or even software security tester roles without huge spatial and temporal constraints. This then suggests role integration (Ashforth et al., 2000) rather than role segmentation (i.e., clear role identities between roles due to low flexibility and permeability of role boundaries). The downside is that role integration creates blurring of role boundaries that instigate cross-role interruptions (i.e., permeable role boundaries), which could create confusion and anxiety (Ashforth et al., 2000) among software developers.

On the other hand, micro role transitions are affected by role identity – a social construct shaped by core and peripheral features (Ashforth et al., 2000). Core features exemplify the main characteristics of a role, while peripheral features signify "secondary" attributes of a role. For example, the core features that shape a software developer role could be logical thinking, creative problem-solving, and team player, whereas the peripheral features could be managerial skills. Difficulty in role transitions is stemmed from the contrast of core and peripheral features between a pair of roles (Ashforth et al., 2000). That is, role transition is affected by the magnitude of changes involved to acquire the skills in the prescribed features of a new role (Ibarra & Barbulescu, 2010). In a DevOps context, software developers may find it hard to transition from developer to systems operator roles, because developers are required to learn a new set of skills related to systems operations.

2.4 Cognitive Load Theory

Practitioners claimed that software developers would prefer software development to systems operations (Shackleford & Kedrosky, 2022), thus suggesting that systems operation is often beyond developers' craft. This then suggests that DevOps poses cognitive challenges (i.e., difficulty in using one's cognitive to address a problem) to software developers. According to Cognitive Load Theory (CLT), human cognition operates through a complex interaction between sensory inputs, short-term memory (i.e., working memory), and long-term memory that serves as a knowledge repository (Sweller, 2010). In comparison to long-term memory and sensory with larger capacity, working memory has relatively limited capacity. Knowledge is retrieved from long-term memory and is then applied to perform tasks using working memory. Cognitive overload occurs when working memory could not process a large volume of information (Sweller, 2010). Alternatively, individuals experience cognitive overload when their cognitive capacities could not meet cognitive demands (Moreno & Mayer, 2007). The following Section 3 addresses cognitive overload among software developers.

2.5 Job Burnout

According to a survey led by Cobalt (i.e., Penetration-as-a-Service vendor), 53% of surveyed developers considered quitting their jobs due to job burnout (Cobalt, 2022). Additionally, Singh et al. (2012) empirically established that software developers' job burnout adversely affected organizational commitment. Benlian (2022) also demonstrated that Agile software development provoked developers' burnout, jeopardizing developers' mental well-being. This suggests that burnout is quite common among software developers. We then argue that software developers who are participating in DevOps would probably experience burnout, as well.

Job burnout affects workers physically (e.g., exhaustion and fatigue) and mentally (e.g., quick to anger, frustration, and sudden irritation) (Freudenberger, 1974). In general, burnout causes exhaustion, cynicism, and deteriorated professional efficacy. Particularly, exhaustion, which exemplifies stress dimension of burnout, causes emotional and cognitive distancing from work. (Maslach et al., 2001). On the other hand, cynicism embodies depersonalization ascribed to indifference attitudes toward work quality; and professional inefficacy entails decline in personal accomplishment and work effectiveness (Maslach et al., 2001). The following Section 3 addresses job burnout among developers.

2.6 Summary of Literature Review

The following table summarizes our literature review.

Key Elements	Descriptions
DevOps	The core value of DevOps lies within the principle of continuity built on CALMS (Hemon-Hildgen & Rowe, 2022). Because technology constantly evolves (Maruping & Matook, 2020), it is necessary to re-examine continuity process (Hemon-Hildgen & Rowe, 2022) from time to time. Therefore, we focus on software security continuity when security is integrated into DevOps.
Role Transitioning Theory	This theory suggests that micro role transition (i.e., role transitions occur with little temporal and spatial constraints) is affected by (1) flexibility and permeability of a role boundary, and (2) role identity (Ashforth et al., 2000). We argue that, for DevOps developers, role transitions involve high flexibility and permeability, primarily because, in an IT working environment, they can easily perform multi-tasking. On the other hand, developers may find it hard to transition from developer to systems operator roles, because they are required to learn a new set of skills related to systems operations. This dichotomy adds an interesting dimension to our research.
Cognitive Load Theory (CLT)	There is a possible cognitive misfit for some DevOps developers who are transitioning into systems operator's role, engendering cognitive challenges. CTL suggests that cognitive challenges would eventually instigate cognitive overload (Sweller, 2010). This study addresses cognitive overload that is possibly caused by role transition.
Job Burnout	Cognitive overload triggers mental exhaustion ascribed to burnout (Schaufeli et al., 1996). We want to address developers' deteriorated mental well-being resulted from burn out, because studies have showed that their well-being affected software quality (Graziotin et al., 2018).

Table 1. Summary of Literature Review

3 Theory Development

3.1 Integrating Security into DevOps

When integrating security into DevOps, it is necessary to collaborate among security, development, and operation teams (Myrbakken & Colomo-Palacios, 2017). Such collaboration occurs at the start of software development, enabling organizations to *shift left* or shift security to the beginning of software development lifecycle (SDLC) (GitLab, 2022). Additionally, speed and agility are critical for DevOps (Rajapakse et al., 2022). This suggests that, not only developers have to be effective in continuous processes including CI (e.g., run error checking and integrate codes built by multiple developers in each development phase) and CD (e.g., deploying new software to a production environment) (Humble & Molesky, 2011), but also they have to effectively run continuous testing (i.e., run automated security test to detect anomaly in each development phase) and monitoring (i.e., consistently produce evidence to show that an application is working properly in each development phase) (Gall & Pigni, 2022; Myrbakken & Colomo-Palacios, 2017). These continuous processes show that software developers transition from one role (e.g., developer's role for code integration in CI) to another (e.g., systems operator's role in continuous monitoring). Embracing a different role would

require developers to apply a different skillset. That is, integrating security into DevOps would require developers play the role of developers, software security testers, and systems operators (GitLab, 2022), in which each role requires a different skillset for tasks completion. Therefore, we argue that, the degree of skill variety, that is, the degree of varied skillsets required to complete a job (Hackman & Oldham, 1976), increases with role transitions in DevOps, especially with security integration. In a similar vein, Tripp et al. (2016) have empirically established that extensive use of Agile practices augmented the degree of skill variety. Based on this rationale, we hypothesize:

H1: In DevOps, role transitioning is positively associated with skill variety

Because developers are taking on systems operator roles defined by systems engineering tasks, developers need to operate in more than one cognitive mode. Specifically, software development embodies divergent thinking, in which a flexible approach of thinking that diverges from the problem on hand (i.e., thinking outside the box) generates a solution (Bishop-Clark, 1995; Gallivan, 2003). In contrast, systems operation generally embraces convergent thinking through which an integrative approach links various systems components to facilitate systems functionalities (Kam & Shang, 2019; McCumber & Sloan, 2002). Rhodes et al. (2008) suggested that a larger scale of systems (e.g., enterprise systems) requires both convergent and divergent thinking to facilitate effective systems engineering and design. Nevertheless, we argue that, in a DevOps context, software developers are mostly responsible for monitoring their software's operations in each IT platform, rather than responsible for designing and administrating the entire enterprise systems. Therefore, software developers' jobs are primarily pertaining to software's operations in a smaller scale, which mainly involve the systems operation's norms of convergent thinking (Lamb & Rhodes, 2008; McCumber & Sloan, 2002). For example, when transitioning from software developer to systems engineer role, it would be challenging for developers to quickly adapt to a different cognitive mode (e.g., convergent thinking) and immediately process a large volume of complex information (e.g., systems configuration and log files) on hand; and as a result, cognitive overload occurs.

Quite often, developers switch frequently between both types of cognitive modes during role transitions (i.e., switching roles between developer and systems operator). This suggests high cognitive demand that would eventually instigate cognitive overload (Kirsh, 2000). Based on this rationale, we propose:

H2: In DevOps, role transitioning is positively associated with cognitive overload

Furthermore, we argue that skill variety has a positive association with cognitive load. As noted, software developers need to switch cognitive gears between divergent thinking and convergent thinking modes during role transitions. Each type of cognitive mode facilitates a unique skillset required for tasks completion in DevOps. Because software developers often manage complexities during software development (Fægri et al., 2010), additional cognitive demands of exercising diverse skillsets (i.e., skill variety) would probably overtax developers' cognitions and trigger cognitive overload (Ju et al., 2021; Vasilescu et al., 2016). Therefore, we theorize:

H3: In DevOps, skill variety is positively associated with cognitive overload

3.2 Job Burnout

With security integration, DevOps requires software developers to obtain different skills in addition to programing skills, because developers wear the hats of systems operators and software security testers (GitLab, 2022). Job Characteristics Theory (JCT) posited that skill variety motivated workers through perceived meaningfulness of their jobs (Hackman & Oldham, 1976). Almost three decades later, Oldham & Hackman (2010) questioned whether the modern social attributes of jobs would replace skill variety. In this context, we argue that DevOps offers a new job dimension, in which the complexity of software development (Fægri et al., 2010; Klemola & Rilling, 2002) and the criticality of time pressure (Maruping et al., 2015) make practicing diverse skills (i.e., skill variety) onerous for

software developers. Along the same line, Tripp et al. (2016) have empirically demonstrated that skill variety in Agile software development did not promote software developers' job satisfaction.

Prior studies proposed that developers gained job satisfaction from creative problem-solving (Gallivan, 2003), rather than from managing complexities required in systems engineering. This suggests that skillsets related to systems engineering (e.g., systems monitoring) defined in a system operator's role may turn out to be a poor cognitive fit or may represent unfamiliar cognitive styles for software developers (Chilton et al., 2005). This would make exercising systems operations difficult (Nelson et al., 2000), thus provoking mental stress attributed to burnout. In addition, software development is complex and non-routine (Fægri et al., 2010), so developers have to exert greater efforts in programming. With high cognitive demands in computer programming (Van Merriënboer & Paas, 1990), using a variety of skills to perform additional tasks would probably provoke work stress that lead to job burnout. We then theorize:

H4: In DevOps, increased skill variety is positively associated with job burnout

As noted earlier, cognitive load pertains to mental efforts put forth by software developers (Gonçales et al., 2021). DevOps requires software developers to play multiple roles and apply different skillsets in each role. This may increase tasks difficulty and demand extra mental efforts, thereby leading to cognitive overload. Prior studies have suggested that cognitive overload is an antecedent to job burnout (Gregory et al., 2017; Hakanen & Bakker, 2017; Iskander, 2019; Vanneste et al., 2021). Built on these prior studies, we hypothesize:

H5: In DevOps, cognitive overload is positively associated with job burnout.

3.3 Software Developers' Continuous Security

As noted, job burnout imposes negative effects on software developers' mental well-being. Prior studies have suggested that burnout would impair cognitive functions such as depletion in cognitive capacity (Van Der Linden et al., 2005). With security integration, DevOps developers are required to actively participate in continuous software security practices encompassing continuous testing, continuous planning, continuous design and development, CI/CD, continuous deployment, and continuous feedback (Kumar & Goyal, 2020). We argue that these activities require high mental attentions and fitness, which are hard to maintain due to job burnout that instigates cognitive impairment. In addition to cognitive deterioration, job burnout provokes indifference attitudes toward job quality and causes decline in professional efficiency (Maslach et al., 2001; Schaufeli et al., 1996). As a result, software developers suffering from job burnout would not be able to focus on continuous software security practices required in DevOps. Based upon this rationale, we propose:

H6: In DevOps, job burnout is negatively associated with software developers' continuous software security practices.

3.4 Automation

Automation encompasses knowledge compilation in that knowledge are organized in a manner that is ready to be applied in a form of task-specific procedures (Van Merriënboer & Paas, 1990). With knowledge compilation, working memory does not have to store and process a large volume of acquired knowledge, because knowledge can be retrieved from and processed by automated processes (Anderson, 1987; Van Merriënboer & Paas, 1990). Accordingly, automation would reduce developers' cognitive demands and attenuates developers' burdens of skill building based upon knowledge acquisition. We then argue that although automation could not completely replace human's decision-making and cognitions, automation helps alleviating the severe impacts of skill variety and cognitive overload on software developers' job burnout. In a similar vein, Hemon-Hildgen et al. (2020) proposed that, in a right working condition, automation could increase developers' job satisfaction when transitioning from Agile software development to DevOps. Additionally, Mueller & Benlian

(2022) empirically demonstrated that automation provided software developers a sense of certainty that alleviated developers' stress. Built on this rationale, we hypothesize:

H7a: In DevOps, automation has a negative moderating effect in the relationship between skill variety and job burnout.

H7b: In DevOps, automation has a negative moderating effect in the relationship between cognitive overloads and job burnout.

Finally, the following diagram depicts our proposed research model.

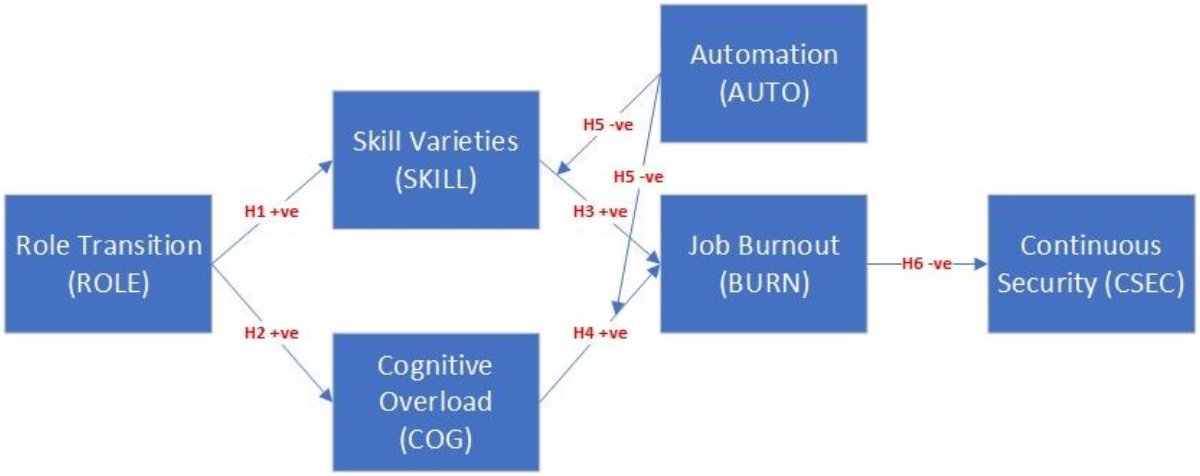

Figure 1. Proposed Research model.

4 Conclusion and Future Research

In the context of Development and Operations (DevOps), this study draws on Role Transitioning Theory (Ashforth et al., 2000) and Cognitive Overload Theory (Sweller, 2010) to investigate software developers' involvement in continuous software security practices. We contend that this study contributes to information systems (IS) security research in several ways. First, to the best of our knowledge, this is one of the few studies that integrates information security into software development. We argue that this would add a new dimension to information systems development (ISD) and enrich IS security research. Second, focusing on software developers' well-being, this study addresses a very important and relevant issue related to software security. Software developers' impacts on software security have often been overlooked, but developers are one of the key actors who are responsible of software security assurance. Therefore, our research findings will provide insights into software security, helping organizations in mitigating software security risks. In the future, this study will design a measurement instrument and later test that instrument using a pilot study. We will also use Qualtrics to recruit software developers who are participating in DevOps. Most likely, we will use Structural Equation Modeling (SEM) for hypotheses testing.

References

Anderson, J. R. (1987). Skill acquisition: Compilation of weak-method problem situations. *Psychological Review, 94*, 192–210. https://doi.org/10.1037/0033-295X.94.2.192

Ashforth, B. E., Kreiner, G. E., & Fugate, M. (2000). All in a Day's Work: Boundaries and Micro Role Transitions. *The Academy of Management Review, 25*(3), 472–491.

August, T., & Tunca, T. I. (2011). Who Should Be Responsible for Software Security? A Comparative Analysis of Liability Policies in Network Environments. *Management Science, 57*(5), 934–959.

Benlian, A. (2022). Sprint Zeal or Sprint Fatigue? The Benefits and Burdens of Agile ISD Practices Use for Developer Well-Being. *Information Systems Research*, *33*(2), 557–578. https://doi.org/10.1287/isre.2021.1069

Bishop-Clark, C. (1995). Cognitive style, personality, and computer programming. *Computers in Human Behavior*, *11*(2), 241–260. https://doi.org/10.1016/0747-5632(94)00034-F

Chilton, M. A., Hardgrave, B. C., & Armstrong, D. J. (2005). Person-Job Cognitive Style Fit for Software Developers: The Effect on Strain and Performance. *Journal of Management Information Systems*, *22*(2), 193–226. https://doi.org/10.1080/07421222.2005.11045849

Cobalt. (2022). *State of Pentesting Report 2022 | Cobalt*. Cobalt Pentest as a Service. https://www.cobalt.io/blog/the-state-of-pentesting-2022-how-labor-shortages-are-impacting-cybersecurity-and-developer-professionals

Deligkaris, P., Panagopoulou, E., Montgomery, A. J., & Masoura, E. (2014). Job burnout and cognitive functioning: A systematic review. *Work & Stress*, *28*(2), 107–123.

Edmundson, C., & Hartman, K. G. (2022). *SANS 2022 DevSecOps Survey: Creating a Culture to Significantly Improve Your Organization's Security Posture*. The SANS Institute. https://www.sans.org/white-papers/sans-2022-devsecops-survey-creating-culture-improve-organization-security/

Fægri, T. E., Dybå, T., & Dingsøyr, T. (2010). Introducing knowledge redundancy practice in software development: Experiences with job rotation in support work. *Information and Software Technology*, *52*(10), 1118–1132. https://doi.org/10.1016/j.infsof.2010.06.002

Freudenberger, H. J. (1974). Staff Burn-Out. *Journal of Social Issues*, *30*(1), 159–165.

Gall, M., & Pigni, F. (2022). Taking DevOps mainstream: A critical review and conceptual framework. *European Journal of Information Systems*, *31*(5), 548–567.

Gallivan, M. J. (2003). The influence of software developers' creative style on their attitudes to and assimilation of a software process innovation. *Information & Management*, *40*(5), 443–465.

GitLab. (2022). *The GitLab 2022 Global DevSecOps Survey* (Thriving in an Insecure World). GitLab. https://about.gitlab.com/developer-survey/#developers

Gonçales, L. J., Farias, K., & da Silva, B. C. (2021). Measuring the cognitive load of software developers: An extended Systematic Mapping Study. *Information and Software Technology*, *136*, 106563. https://doi.org/10.1016/j.infsof.2021.106563

Graziotin, D., Fagerholm, F., Wang, X., & Abrahamsson, P. (2018). What happens when software developers are (un)happy. *Journal of Systems and Software*, *140*, 32–47.

Graziotin, D., Wang, X., & Abrahamsson, P. (2014). Happy software developers solve problems better: Psychological measurements in empirical software engineering. *PeerJ*, *2*, e289.

Gregory, M. E., Russo, E., & Singh, H. (2017). Electronic Health Record Alert-Related Workload as a Predictor of Burnout in Primary Care Providers. *Applied Clinical Informatics*, *08*(3), 686–697.

Hackman, J. R., & Oldham, G. R. (1976). Motivation through the design of work: Test of a theory. *Organizational Behavior and Human Performance*, *16*(2), 250–279.

Hakanen, J. J., & Bakker, A. B. (2017). Born and bred to burn out: A life-course view and reflections on job burnout. *Journal of Occupational Health Psychology*, *22*(3), 354–364.

Hall, D. T., & Richter, J. (1988). Balancing Work Life and Home Life: What Can Organizations Do to Help? *Academy of Management Perspectives*, *2*(3), 213–223.

Hemon, A., Lyonnet, B., Rowe, F., & Fitzgerald, B. (2020). From Agile to DevOps: Smart Skills and Collaborations. *Information Systems Frontiers*, *22*(4), 927–945.

Hemon-Hildgen, A., & Rowe, F. (2022). Conceptualising and defining DevOps: A review for understanding, not a framework for practitioners. *European Journal of Information Systems*, *31*(5), 568–574. https://doi.org/10.1080/0960085X.2022.2100061

Hemon-Hildgen, A., Rowe, F., & Monnier-Senicourt, L. (2020). Orchestrating automation and sharing in DevOps teams: A revelatory case of job satisfaction factors, risk and work conditions. *European Journal of Information Systems*, *29*(5), 474–499. https://doi.org/10.1080/0960085X.2020.1782276

Humble, J., & Molesky, J. (2011). Why enterprises must adopt devops to enable continuous delivery. *Cutter IT Journal*, *24*(8), 6.

Hüttermann, M. (2021). The DevOps Continuum: Walking the Shadowy Bridge from Information Systems Development to Operations. *ECIS 2021 Research Papers*. https://aisel.aisnet.org/ecis2021_rp/78

Ibarra, H., & Barbulescu, R. (2010). Identity as Narrative: Prevalence, Effectiveness, and Consequences of Narrative Identity Work in Macro Work Role Transitions. *The Academy of Management Review*, *35*(1), 135–154.

IBM Cloud Education. (2020). *What is DevSecOps?* IBM. https://www.ibm.com/cloud/learn/devsecops

Iskander, M. (2019). Burnout, Cognitive Overload, and Metacognition in Medicine. *Medical Science Educator*, *29*(1), 325–328. https://doi.org/10.1007/s40670-018-00654-5

Ju, A., Sajnani, H., Kelly, S., & Herzig, K. (2021). A Case Study of Onboarding in Software Teams: Tasks and Strategies. *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 613–623. https://doi.org/10.1109/ICSE43902.2021.00063

Kam, H.-J., & Shang, Y. (2019, June 15). Improving Cybersecurity Learning: An Integration of Cyber Offense and Cyber Defense. *PACIS 2019 Proceedings*. The Pacific Asia Conference on Information Systems, Xi'an, China. https://aisel.aisnet.org/pacis2019/177

Kirsh, D. (2000). A Few Thoughts on Cognitive Overload. *Intellectica*, *1*(30), 19–51.

Klemola, T., & Rilling, J. (2002). Modeling comprehension processes in software development. *Proceedings First IEEE International Conference on Cognitive Informatics*, 329–336.

Kossiakoff, A., Biemer, S. M., Seymour, S. J., & Flanigan, D. A. (2020). *Systems Engineering Principles and Practice*. John Wiley & Sons.

Kumar, R., & Goyal, R. (2020). Modeling continuous security: A conceptual model for automated DevSecOps using open-source software over cloud (ADOC). *Computers & Security*, *97*, 101967.

Maruping, L. M., & Matook, S. (2020). The evolution of software development orchestration: Current state and an agenda for future research. *European Journal of Information Systems*, *29*(5), 443–457.

Maruping, L. M., Venkatesh, V., Thatcher, S. M. B., & Patel, P. C. (2015). Folding Under Pressure or Rising to the Occasion? Perceived Time Pressure and the Moderating Role of Team Temporal Leadership. *Academy of Management Journal*, *58*(5), 1313–1333.

Maslach, C., Schaufeli, W. B., & Leiter, M. P. (2001). Job Burnout. *Annual Review of Psychology*, *52*(1), 397–422. https://doi.org/10.1146/annurev.psych.52.1.397

Matook, S., Lee, G., & Fitzgerald, B. (2021). MISQ Research Curation on Information Systems Development. *MIS Quarterly*.

McCumber, W. H., & Sloan, C. (2002). Educating Systems Engineers: Encouraging Divergent Thinking. *INCOSE International Symposium*, *12*(1), 8–15.

Moreno, R., & Mayer, R. (2007). Interactive Multimodal Learning Environments: Special Issue on Interactive Learning Environments: Contemporary Issues and Trends. *Educational Psychology Review*, *19*(3), 309–326. https://doi.org/10.1007/s10648-007-9047-2

Mueller, L., & Benlian, A. (2022). Too Drained from Being Agile? The Self-Regulatory Effects of Agile ISD Practices Use and their Consequences for Turnover Intention. *JAIS Preprints (Forthcoming)*. https://doi.org/10.17705/1jais.00766

Myrbakken, H., & Colomo-Palacios, R. (2017). DevSecOps: A Multivocal Literature Review. In A. Mas, A. Mesquida, R. V. O'Connor, T. Rout, & A. Dorling (Eds.), *Software Process Improvement and Capability Determination* (pp. 17–29). Springer International Publishing.

Nelson, K. M., Nadkarni, S., Narayanan, V. K., & Ghods, M. (2000). Understanding Software Operations Support Expertise: A Revealed Causal Mapping Approach. *MIS Quarterly*, *24*(3), 475–507. https://doi.org/10.2307/3250971

Oldham, G. R., & Hackman, J. R. (2010). Not what it was and not what it will be: The future of job design research. *Journal of Organizational Behavior*, *31*(2–3), 463–479.

Oosterholt, B. G., Van der Linden, D., Maes, J. H., Verbraak, M. J., & Kompier, M. A. (2012). Burned out cognition—Cognitive functioning of burnout patients before and after a period with psychological treatment. *Scandinavian Journal of Work, Environment & Health*, *38*(4), 358–369.

Pleck, J. H. (1977). The Work-Family Role System*. *Social Problems*, *24*(4), 417–427.

Rajapakse, R. N., Zahedi, M., Babar, M. A., & Shen, H. (2022). Challenges and solutions when adopting DevSecOps: A systematic review. *Information and Software Technology*, *141*, 106700.

Rasch, R. H., & Tosi, H. L. (1992). Factors Affecting Software Developers' Performance: An Integrated Approach. *MIS Quarterly*, *16*(3), 395–413. https://doi.org/10.2307/249535

Rezvani, A., & Khosravi, P. (2019). Emotional intelligence: The key to mitigating stress and fostering trust among software developers working on information system projects. *International Journal of Information Management*, *48*, 139–150. https://doi.org/10.1016/j.ijinfomgt.2019.02.007

Richter, J. (1984). *The daily transition between professional and private life*. Boston University.

Rizzo, J. R., House, R. J., & Lirtzman, S. I. (1970). Role Conflict and Ambiguity in Complex Organizations. *Administrative Science Quarterly*, *15*(2), 150–163. https://doi.org/10.2307/2391486

Schaufeli, W., Leiter, M., Maslach, C., & Jackson, S. (1996). Maslach Burnout Inventory—General Survey (GS). In C. Maslach, S. Jackson, & M. P. Leiter (Eds.), *The Maslach Burnout Inventory: Test manual* (Vol. 31). Consulting Psychologists Press.

Shackleford, D., & Kedrosky, E. (Directors). (2022, March 24). *DevSecOps: Winning Principles for Security in DevOps*. https://www.sans.org/webcasts/devsecops-winning-principles-for-security-in-devops/

Siavvas, M., Tsoukalas, D., Jankovic, M., Kehagias, D., & Tzovaras, D. (2022). Technical debt as an indicator of software security risk: A machine learning approach for software development enterprises. *Enterprise Information Systems*, *16*(5), 1824017.

Silverthorne, V. (2022, August 31). Why—And how—DevOps roles are changing. *GitLab*. https://about.gitlab.com/blog/2022/08/31/the-changing-roles-in-devsecops/

Singh, P., Suar, D., & Leiter, M. P. (2012). Antecedents, Work-Related Consequences, and Buffers of Job Burnout Among Indian Software Developers. *Journal of Leadership & Organizational Studies*, *19*(1), 83–104. https://doi.org/10.1177/1548051811429572

Sweller, J. (2010). Element Interactivity and Intrinsic, Extraneous, and Germane Cognitive Load. *Educational Psychology Review*, *22*(2), 123–138. https://doi.org/10.1007/s10648-010-9128-5

Tripp, J., Riemenschneider, C., & Thatcher, J. (2016). Job Satisfaction in Agile Development Teams: Agile Development as Work Redesign. *Journal of the Association for Information Systems*, *17*(4).

Van Der Linden, D., Keijsers, G. P. J., Eling, P., & Schaijk, R. V. (2005). Work stress and attentional difficulties: An initial study on burnout and cognitive failures. *Work & Stress*, *19*(1), 23–36.

Van Merriënboer, J. J. G., & Paas, F. G. W. C. (1990). Automation and schema acquisition in learning elementary computer programming: Implications for the design of practice. *Computers in Human Behavior*, *6*(3), 273–289. https://doi.org/10.1016/0747-5632(90)90023-A

Vanneste, P., Raes, A., Morton, J., Bombeke, K., Van Acker, B. B., Larmuseau, C., Depaepe, F., & Van den Noortgate, W. (2021). Towards measuring cognitive load through multimodal physiological data. *Cognition, Technology & Work*, *23*(3), 567–585.

Vasilescu, B., Blincoe, K., Xuan, Q., Casalnuovo, C., Damian, D., Devanbu, P., & Filkov, V. (2016). The sky is not the limit: Multitasking across GitHub projects. *Proceedings of the 38th International Conference on Software Engineering*, 994–1005. https://doi.org/10.1145/2884781.2884875

Venkatesh, V., Thong, J. Y. L., Chan, F. K. Y., Hoehle, H., & Spohrer, K. (2020). How agile software development methods reduce work exhaustion: Insights on role perceptions and organizational skills. *Information Systems Journal*, *30*(4), 733–761. https://doi.org/10.1111/isj.12282

Verizon. (2022). *2022 Data Breach Investigations Report (DBIR)*. https://www.verizon.com/business/resources/reports/dbir/2022/master-guide/

Wiedemann, A., Wiesche, M., Gewald, H., & Krcmar, H. (2020). Understanding how DevOps aligns development and operations: A tripartite model of intra-IT alignment. *European Journal of Information Systems*, *29*(5), 458–473. https://doi.org/10.1080/0960085X.2020.1782277

Windeler, J. B., Maruping, L., & Venkatesh, V. (2017). Technical Systems Development Risk Factors: The Role of Empowering Leadership in Lowering Developers' Stress. *Information Systems Research*, *28*(4), 775–796. https://doi.org/10.1287/isre.2017.0716

Yasin, A., Liu, L., Li, T., Fatima, R., & Jianmin, W. (2019). Improving software security awareness using a serious game. *IET Software*, *13*(2), 159–169. https://doi.org/10.1049/iet-sen.2018.5095