Individual Differences in Trust in Code: The Moderating Effects of Personality on the Trustworthiness-Trust Relationship

Tyler J. Ryan[1], Charles Walter[2], Gene M. Alarcon[3(✉)], Rose F. Gamble[2], Sarah A. Jessup[3], and August A. Capiola[4]

[1] SRA International, Inc., A CSRA Company, Dayton, OH 45431, USA
tyler.ryan@csra.com
[2] University of Tulsa, Tulsa, OK 74104, USA
[3] U.S. Air Force Research Laboratory, 711th HPW RHXS, WPAFB, Dayton, OH, USA
gene.alarcon.l@us.af.mil
[4] Consortium Research Fellows Program, Lancaster, OH, USA

Abstract. The daily use of technology has made people ever more reliant on software. It is important these software systems are produced in a manner that is both efficient and secure. In this context, psychological trust of software is a pertinent aspect of research. The present study explored the relationship of trustworthiness ratings, propensity to trust, and trait suspicion on software reuse. In addition, we explored personality as a moderator of the trustworthiness-reuse relationship, as hypothesized in the interpersonal trust literature [1]. We recruited participants from Amazon's Mechanical Turk and requested they assess classes of Java code. Analyses revealed trait suspicion influenced decisions to reuse code and moderated the trustworthiness-trust relationship. A dual-process model of information processing was adopted for interpretation of these effects. Implications include contributions to research and theory on psychological trust, as well as practical implications for personnel selection with regard to software production.

Keywords: Trust · Suspicion · Software reuse

1 Introduction

The integration of software systems into nearly every aspect of modern life has increased productivity, but also vulnerability. Our reliance on these systems creates a need for safe and secure software produced in a timely manner. This, in turn, has led to a need for reusing software. However, little is known about how developers perceive and comprehend code. Designing software systems with the human user in mind requires an understanding of the factors involved in human decision-making. The human-computer interaction literature emphasizes that systems should not be relied upon beyond their capabilities [2]. Psychological theories of trust can help inform software development and reuse practices by understanding the antecedents to software reuse.

C. Stephanidis (Ed.): HCII Posters 2018, CCIS 850, pp. 370–376, 2018.
https://doi.org/10.1007/978-3-319-92270-6_53

1.1 Software Reuse and Review

Modern software production relies on source code reuse practices [3]. Frakes and Kang [3] defined software reuse as, "the use of existing software or software knowledge to construct new software," (p. 529). Reusing software increases productivity by reducing development time [4]. Systematic reuse practices can even lead to fewer software defects in the source code [5]. However, reuse practices can lead to decreased reliability and increased vulnerability if software is not properly vetted. Additionally, integrating components written by different developers into a single project requires some degree of trust in the software. Developers can choose to review code in great detail, but this is less time-efficient. Instead, developers may perform a cursory review of the code to make a decision.

1.2 Trust and Software Reuse

Trust is defined as the willingness to make oneself vulnerable to another [1]. Although this definition was created for describing the interpersonal trust process, the definition has also been applied to trust in automation contexts [2]. Recent research has investigated how software characteristics can influence psychological trust in software systems and source code [6–8]. In this context, trust is the willingness to reuse the code [6]. Reusing code makes one vulnerable, as the code may contain defects. A recent cognitive task analysis indicated that the software, the environment, and individual differences in the developer can influence user perceptions of software trustworthiness [6].

1.3 Heuristic Systematic Processing Model of Code

Researchers [7, 8] have developed a model of trust in software systems based on the heuristic-systematic model (HSM) of information processing [9, 10]. The HSM was originally developed as a model of psychological persuasion and presents two primary ways to process information in a message, namely heuristic and systematic processing [9]. Heuristic processing entails mental shortcuts that serve as sufficient justifications for accepting arguments [10]. In the programming context, coders may view easily identifiable code attributes (e.g., source of the code). These attributes serve as cues, which influence a developer's decision to reuse [7]. In turn, reuse can be attributed to heuristic processing of the source cue. Systematic processing involves a detailed analysis, requiring significantly more cognitive processing [9]. The sufficiency principle in the HSM states that perceivers attempt to strike a balance between conserving cognitive resources and obtaining sufficient confidence in their assessments [10]. The sufficiency threshold is the minimum level of desired confidence about one's judgements before making a decision. When the sufficiency threshold is low, little information is needed for a decision. When the sufficiency threshold is high, more information is needed and systematic processing is typically employed. Individual differences can influence one's sufficiency threshold [7]. Some developers may tend towards heuristic processing while others may tend towards systematic processing. Researchers [7, 8] have suggested personality may influence developer's information processing strategies, leading to different reuse outcomes.

1.4 Personality, Trust, and Software Review

Personality is the "characteristic patterns of thought, emotion, and behavior together with the psychological mechanisms – hidden or not – behind those patterns" [11] and is a topic of interest in the computer science literature [12]. Studies have focused on the influence of personality on pair programming [13], team climate, and performance [14]. Although research has explored the effects of personality on software development, no research has explored how personality affects psychological trust in software systems. Trust is based on the individual's perceptions of the referent from information in the environment (i.e., trustworthiness) [1, 15]. Trust can be based on one's general propensity to trust [1] or situation specific information (e.g., interacting with a referent) [15], such as aspects of the software [6]. According to Mayer et al. [1], perceptions of trustworthiness affect decisions to trust, and this in turn affects outcomes of trust (i.e., actual reuse of code). Therefore, we hypothesize that perceived trustworthiness positively relates to trust intentions (H_1).

Trust perceptions are not objective. Individual differences in the trustor can affect their willingness to trust the referent, as well as the relationship between perceived trustworthiness and trust [1]. In other words, individual differences have been hypothesized to have direct and moderating influences on trust intentions [1, 6]. However, no research to date has explored these effects on code reuse (i.e. trust intentions). We explored propensity to trust [1], or PT, and trait suspicion [16] as individual differences that would have direct and indirect effects on reuse. PT is the disposition to trust others across contexts [1]. In contrast, trait suspicion [16] is characterized by uncertainty, perceived mal-intent, and cognitive activity. Participants that have a tendency to trust others should have a higher likelihood of reusing code. We hypothesize that PT has a positive direct effect on trust intentions (H_2). In contrast, those who have a propensity to be suspicious of others will be less likely to reuse code, as they are inclined to seek information and be wary of potential harm. Therefore, we hypothesize that trait suspicion has a negative direct effect on trust intentions (H_3).

Trust perceptions can be influenced by individual differences [1]. Individuals high in PT may have lower sufficiency thresholds. PT may moderate the trustworthiness-trust relationship, as the information gained from the code should be relied on to a greater extent. We hypothesize that PT has a moderating effect on the relationship between trustworthiness perceptions and trust in code (H_4). In contrast, individuals higher on the cognitive activity dimension of suspicion should have a higher sufficiency threshold and require more information about the referent to make a decision. Trustworthiness ratings comprise perceptions of a referent. As such, the cognitive activity dimension of trait suspicion should affect how relevant trustworthiness perceptions are for informing trust. Therefore, we hypothesize that trait suspicion moderates the relationship between perceived trustworthiness and trust intentions (H_5).

2 Method

Subjects were recruited using Amazon's Mechanical Turk (Mturk). Participants were required to have at least 3 years of coding experience and have a working knowledge of Java. A total of 45 participants were recruited. The final sample consisted of 11 (24.4%)

females and 34 (75.5%) males, with a mean age of 29.13 years (SD = 6.57), and a mean of 6.69 (SD = 4.99) years of programming experience, respectively. Participants received 10.00 USD for participation in the study, which was paid through Mturk's worker payment system.

2.1 Measures

Trustworthiness and Trust Assessments
Perceived trustworthiness was measured with a single item asking participants, "How trustworthy is the code?" Participants responded using a 1 to 7 Likert-type scale (1 = strongly untrustworthy, 7 = strongly trustworthy). Trust in the code was measured with a single item asking participants whether they would use the code or not.

Personality Measures
Propensity to trust (PT) was measured with the eight item Propensity to Trust scale [17], which contains items measuring beliefs about everyday phenomena. Participants responded on a Likert-type scale (1 = strongly disagree, 5 = strongly agree), α = .66. The Suspicion Propensity Index – I [16] was used to measure trait suspicion. Participants evaluate eleven fictional vignettes on four Likert-type sub-scales (1 = strongly disagree, 5 = strongly agree), two of which we implemented in the present study: (1) uncertainty and mal-intent, α = .78, measuring one's propensity to view the motivations of others as malicious, and (2) uncertainty and cognitive activity, α = .73, measuring one's propensity to seek out and reflect on information about a referent they are unsure about. Trait suspicion scores were calculated as the sum of the two facets, as recommended by Calhoun et al. [16].

2.2 Stimuli

Stimuli consisted of images of Java code artifacts which were taken from publically available open source Java code repositories on GitHub.com. Each artifact selected from the overall code. Artifacts were complex enough to require review to fully understand, but simple enough to be reviewed within approximately 10 min by an experienced reviewer. Artifacts had 4.4% to 78.3% of their lines commented. The commenting percentage range is reflective of the expectations Java programmers have when reviewing code samples. Some samples need very few comments for large, straight-forward sections, while other samples need long, multi-line comments for more complex, possibly smaller, code segments. The artifacts were cleaned by removing existing or references to authorship, which could influence participant trust. Cleaning resulted in the removal or rewriting of unusual comments, as well as adjusting the code in accordance with Java style guides [18]. This allowed us to separate the most common commenting errors into three categories: style, validity, and placement.

The artifacts were then reviewed by outside reviewers to ensure that no additional errors were present. The artifacts were separated such that there were two code samples, each with minimal or heavily degraded style, validity, and placement. Style degradations resulted in adding in old code which was commented out, commenting only some of the defined functions, or not using proper Java style for long comments.

Validity degradations included adding new or modifying existing comments to be incorrect (e.g., possibly indicating a code change or multiple editors), to lack useful information, to include information relating to code that needs to be added, and to be irrelevant. Placement degradations altered comments in ways that were against Java conventions and included comments which were overly verbose for simple concepts.

Each artifact was displayed to participants through an online portal that collected their responses to the trust and personality assessment questions. All stimuli were presented to the participants in the form of images of the code, with syntax highlighting consistent with the default highlighting of Eclipse, a widely used Java development program.

2.3 Procedure

Participants were administered the initial surveys and then provided written instructions for the upcoming task. Each participant was then given 18 different Java classes ranging between 29 and 390 lines of code. Participants were asked to evaluate their perceived trustworthiness of each Java class and then decide whether they would use the code. Upon completion of the code evaluation task, participants were provided remuneration for their time.

3 Analysis and Results

A generalized estimating equation approach was used for the primary analysis to account for the repeated measures design and model the binary outcomes [19]. A model building approach was used, successively adding and selecting those variables that improved model fit using Wald χ^2 tests and comparing Quasi-information criterion statistics (QIC) [20]. An exchangeable correlation structure was chosen to model the residual correlation between the repeated measures.

An initial null model was run with only an intercept as a predictor to compare successive models, QIC = 935.90. Model 1 consisted of the main effects of the four factors, style, validity, placement, and order of stimulus presentation. None of the main effects were significant, and the factors did not provide significant explanation of the variance in reuse over the null model, Wald χ^2 (4) = 3.24, p = .520, QIC = 941.00, and were excluded from further analyses. Model 2 regressed trust intentions onto perceived trustworthiness. Trustworthiness, Wald χ^2 (1) = 132, $p < .001$, QIC = 373.88 was a significant predictor of reuse intentions supporting H_1. Model 3 added PT as a covariate to the model. The addition of PT was only marginally significant, Wald χ^2 (1) = 2.81, p = .093, QIC = 373.64, failing to support H_2. PT was thus dropped from further analyses, and H_4 was not tested. Model 4 added trait suspicion as a main effect with trustworthiness. Trait suspicion significantly increased the amount of variance explained in use, Wald χ^2 (1) = 7.22, p = .007, QIC = 369.76, supporting H_3. Model 5 included the main effects of trustworthiness and trait suspicion, along with their interaction. The inclusion of the interaction term significantly changed the model, Wald χ^2 (1) = 6.50, p = .011, QIC = 366.12, supporting H_5. The intercept for the model was not significant, β = 1.75, χ^2 (1) = 0.31, p = .579. Trustworthiness as a

main effect was also not significant, $\beta = 0.08$, χ^2 (1) = 0.01, $p = .912$. The main effect of trait suspicion had a significant negative influence on reuse intentions, $\beta = -1.79$, χ^2 (1) = 8.65, $p = .003$. The main effect was qualified with a two-way interaction. The interaction between trustworthiness and trait suspicion was positively significant, $\beta = 0.38$, χ^2 (1) = 6.50, $p = .011$.

4 Discussion and Conclusion

The current study explored trustworthiness perceptions, personality, and the interaction of the two as predictors of trust (i.e., code reuse). As expected, trustworthiness perceptions accounted for significant variance in code reuse. PT was not a significant predictor. These findings may be due to the scale focusing on the general beliefs about others trustworthiness, rather than computer mediated behaviors such as code review. Suspicion propensity had a significant negative effect on use intentions. When reviewing software, a developer may not yet understand the code, inducing uncertainty with regard to its trustworthiness and reliability. Developers that tend to address this uncertainty by seeking out and processing information about the code, while also perceiving the potential for malicious intentions in the code, will be less likely to reuse the code. The current study also found an interaction effect between trustworthiness perceptions and trait suspicion. The effect was positive, suggesting that trait suspicion strengthens the relationship between perceived trustworthiness and trust. From an information processing perspective, results indicate that trustworthiness perceptions are trust-relevant information about a referent that the trustee can use to inform their trust decisions. Those that are naturally higher in suspicion, and therefore cognitive activation, should have a higher propensity to systematically process information in the face of uncertainty. These results also support Mayer et al.'s [1] hypothesis that individual differences moderate the trustworthiness-trust relationship. The study contributes to the current literature by providing new insight for the trust process. The study also suggests that an understanding of a developer's personality may be useful for personnel selection for software development teams.

References

1. Mayer, R.C., Davis, J.H., Schoorman, F.D.: An integrative model of organizational trust. Acad. Manag. Rev. **20**, 709–734 (1995). https://doi.org/10.5465/AMR.1995.9508080335
2. Lee, J.D., See, K.A.: Trust in automation: designing for appropriate reliance. Hum. Factors **46**, 50–80 (2004). https://doi.org/10.1518/hfes.46.1.50_30392
3. Frakes, W.B., Kang, K.: Software reuse research: status and future. IEEE Trans. Softw. Eng. **31**, 529–536 (2005). https://doi.org/10.1109/TSE.2005.85
4. Banker, R.D., Kauffman, R.J.: Reuse and productivity in integrated computer-aided software engineering: an empirical study. MIS Q. **14**, 375–401 (1991). https://doi.org/10.2307/249649
5. Lim, W.C.: Effects of reuse on quality, productivity, and economics. IEEE Softw. **11**, 23–30 (1994). https://doi.org/10.1109/52.311048

6. Alarcon, G.M., Militello, L.G., Ryan, P., Jessup, S.A., Calhoun, C.S., Lyons, J.B.: A descriptive model of computer code trustworthiness. J. Cognit. Eng. Decis. Mak. **11**, 107–121 (2017). https://doi.org/10.1177/1555343416657236
7. Alarcon, G.M., Ryan, T.J.: Trustworthiness perceptions of computer code: a heuristic-systematic processing model. In: Proceedings of 51st Hawaii International Conference on System Sciences, pp. 5384–5393 (2018). http://hdl.handle.net/10125/50560
8. Alarcon, G.M., Gamble, R., Jessup, S.A., Walter, C., Ryan, T.J., Wood, D.W., Calhoun, C.: Application of the heuristic-systematic model to computer code trustworthiness: the influence of reputation and transparency. Cogent Psychol. **4**, 1389640 (2017). https://doi.org/10.1080/23311908.2017.1389640
9. Chaiken, S.: Heuristic versus systematic information processing and the use of source versus message cues in persuasion. J. Pers. Soc. Psychol. **39**, 752–766 (1980). https://doi.org/10.1037/0022-3514.39.5.752
10. Chen, S., Duckworth, K., Chaiken, S.: Motivated heuristic and systematic processing. Psychol. Inq. **10**, 44–49 (1999). https://doi.org/10.1207/s15327965pli1001_6
11. Funder, D.C.: The Personality Puzzle. WW Norton & Co, New York (1997)
12. Cruz, S., da Silva, F.Q., Capretz, L.F.: Forty years of research on personality in software engineering: a mapping study. Comput. Hum. Behav. **46**, 94–113 (2015). https://doi.org/10.1016/j.chb.2014.12.008
13. Sfetsos, P., Stamelos, I., Angelis, L., Deligiannis, I.: An experimental investigation of personality types impact on pair effectiveness in pair programming. Empirical Softw. Eng. **14**, 187–266 (2009). https://doi.org/10.1007/s10664-008-9093-5
14. Soomro, A.B., Salleh, N., Mendes, E., Grundy, J., Burch, G., Nordin, A.: The effect of software engineers' personality traits on team climate and performance: a systematic literature review. Inf. Softw. Technol. **73**, 52–65 (2016). https://doi.org/10.1016/j.infsof.2016.01.006
15. Jones, S.L., Shah, P.P.: Diagnosing the locus of trust: a temporal perspective for trustor, trustee, and dyadic influences on perceived trustworthiness. J. Appl. Psychol. **101**, 392–414 (2016). https://doi.org/10.1037/apl0000041
16. Calhoun, C., Bobko, P., Schuelke, M., Jessup, S., Ryan, T., Walter, C., Gamble, R.F., Hirshfield, L., Bowling, N., Bragg, C., Khazon, S.: Suspicion, Trust, and Automation (SRA International Inc. Publication No. AFRL-RH-WP-TR-2017-0002) (2017). https://pdfs.semanticscholar.org/de09/24c9e525b8003a28c75eec1554baacb6d15a.pdf
17. Mayer, R.C., Davis, J.H.: The effect of the performance appraisal system on trust for management: a field quasi-experiment. J. Appl. Psychol. **84**, 123–136 (1999). https://doi.org/10.1037/0021-9010.84.1.123
18. Google: Java Style Guidelines (2014). https://google.github.io/styleguide/javaguide.html
19. Zeger, S.L., Liang, K.Y., Albert, P.S.: Models for longitudinal data: a generalized estimating equation approach. Biometrics **44**, 1049–1060 (1988). https://doi.org/10.2307/2531734
20. Pan, W.: Akaike's information criterion in generalized estimating equations. Biometrics **57**, 120–125 (2001). https://doi.org/10.1111/j.0006-341X.2001.00120.x