Application of the heuristic-systematic model to computer code trustworthiness: The influence of reputation and transparency

Gene M. Alarcon1*, Rose Gamble2, Sarah A. Jessup3, Charles Walter2, Tyler J. Ryan1, David W. Wood3 and Chris S. Calhoun3

Abstract: Computer programs (code) are integral to the functions of current society. Yet, little is known about why programmers trust code they did not create. The current paper applied the heuristic-systematic model (HSM) of information processing to perceptions of code trust for reuse. The studies explored transparency (readability and organization) and reputation (source) as factors that influenced trust perceptions and time spent reviewing code using professional programmers. Source and readability manipulations led to higher trustworthiness assessments in the first study. Organization had nonlinear effects on trustworthiness. A three-way interaction including time was also found. The second online study largely replicated the first study’s main and interaction effects for trustworthiness, but the main effects on time were not significant. Our findings suggest the relationships of transparency on trustworthiness are not as straightforward as previously thought. Additionally, the findings of the current study expand the HSM to trust in code.

1. Introduction

Computer programs, also known as software or simply code, have become ubiquitous in the current technological society. Almost every aspect of our lives has a digital component with code running behind the scenes, such as online banking, power grids, cell phones, traffic signals, and medical equipment. The pervasiveness of code has led to reliance on integrating code from multiple sources and deploying code patches or updates in a timely manner, leading to a large increase in code reuse, where programmers integrate acquired code that they did not write. A recent article by CNET estimated that approximately 80–90% of code has been reused by programmers (Hautala, 2015). Code
reuse indicates a perception that the code is trustworthy, as the programmer might use the code in an application for which it was not originally intended. Code reuse can be advantageous, but it can also present possible risks. For example, the OpenSSL encryption was widely used by banks for online security. The pervasive reuse of the code demonstrated perceived trustworthiness. However, Google’s (2015) security team identified a serious issue in the software, known as heartbleed (CVE Mitre, 2014; Durumeric et al., 2014; Grubb, 2014) that resulted in the theft of financial and personal data. The widespread use of code and the risks associated with decisions to reuse code make perceived trustworthiness of code an important aspect of psychological research that has rarely been explored.

Code is a collection of computer instructions written using a computer language as text (Harman, 2010). Although code can be created entirely by an individual, large software projects often employ tactics to reuse code for efficiency. Code may be created in teams where one aspect of code is created by a team member and then passed along to another team member. Code may also be downloaded from repositories, which have code that was created by communities, for use in larger projects. The computer science industry has suggested conventions for creating code that assists in code reuse (Allamanis, Barr, Bird, & Sutton, 2014). The industry also employs static analysis tools to identify bugs, security threats, and other issues in code that are defined by the organization within the rules of the tools. Trust is an integral part of these processes. Although there is a consensus that trustworthiness and trust in code are important research questions, little research to date has explored these relationships effectively. Psychological theories might help explicate the relationship between the programmer and the code, allowing the mapping of accepted trustworthiness factors for program reuse and integration into larger systems.

No theories of trust in code exist to the knowledge of the authors. Instead, we rely on theories from the trust in automation and interpersonal trust literature. Trust is the willingness to be vulnerable to some referent. Trust can be separated into three aspects: trust beliefs, trust intentions, and trust actions (Jones & Shah, 2016). The focus of the current article is on beliefs and actions. Trust beliefs are perceptions of a referent (perceived trustworthiness of the code). It is important to note trust beliefs are not always factual and thus may be inaccurate. Trust actions are the behavioral reliance on the referent, such as reusing a piece of code. Trust has traditionally been thought of as an interpersonal construct. However, the literature has expanded trust to organizations (Mayer, Davis, & Schoorman, 1995), automation (Hoff & Bashir, 2015), human–machine teaming (de Visser & Parasuraman, 2011), and websites (Flavián & Guinalíu, 2006). We further expand the trust literature to perceptions of code.

2. Heuristic-systematic processing model
The heuristic-systematic model (HSM) of information processing is a model that explains how people are persuaded by messages (Chen, Duckworth, & Chaiken, 1999). The model has been applied to persuasion concepts such as consumer purchasing decisions (Darke, Freedman, & Chaiken, 1995) and politics (Griffin, Neuwirth, Giese, & Dunwoody, 2002). Also, the model may add insight into how a programmer perceives computer code. The HSM states there are two paths that influence persuasion: heuristic processing and systematic processing. Heuristic processing involves retrieving task-relevant rules or “heuristics” stored in memory. A heuristic is a “strategy that ignores part of the information, with the goal of making decisions quickly” (p. 454) than more intricate strategies (Gigerenzer & Gaissmaier, 2011). Heuristics are learned knowledge structures (e.g. biases, norms, established procedures, rules of thumb; Chaiken, Liberman, & Eagly, 1989). For example, a voter that uses their political party as the sole basis for voting employs heuristic processing. Systematic processing involves a critical examination of pertinent information to form a decision (Chen et al., 1999). A voter that has no political affiliation, such as an independent, compares and contrasts information about the candidates using systematic processing. Judgments based on heuristic processing take less cognitive effort and time but may lead to less accuracy (Chen et al., 1999). In contrast, systematic processing is purpose driven and takes considerably more time and cognitive effort to process information, leading to more accurate assessments. For example, Chaiken (1980) found that participants with greater accuracy concern and involvement in reading persuasive arguments took more time to read, indicating a systematic approach, compared to those with less involvement and
concern. In the HSM, the two processes are not always independent and can occur at the same time, leading to co-processing of information. An example of co-processing in politics is a voter who may have a political affiliation or leaning (heuristic) but the individual compares and contrasts the information about each candidate before making a decision (systematic).

The HSM states that perceivers are economy driven, attempting to use the least effort to process the information (Fiske & Taylor, 1991). Perceivers attempt to balance minimizing effort and maximizing confidence. However, efficiency is not the only motivation in the model. The sufficiency principle of the HSM states that a perceiver has an actual level of confidence and a desired level of confidence (Chen et al., 1999). The sufficiency threshold is when the level of confidence will satisfy the motives the perceiver is trying to meet. If the sufficiency threshold is met, the perceiver discontinues processing and makes a decision. If the threshold is not met, the perceiver will continue processing. The sufficiency threshold is not static and can change given the motivation of the perceiver (Chaiken, Giner-Sorolla, & Chen, 1996). Three motivations influence the sufficiency threshold: accuracy motivation, defense motivation, and impression motivation (Chaiken et al., 1996). We propose the motivation most relevant to programmers is accuracy motivation. Accuracy motivation is the “open-minded and evenhanded judgment of relevant information” (Chen et al., 1999, p. 45). Accuracy motivation is driven by a need to precisely evaluate the referent or message without making any mistakes. If cognitive resources or accuracy motivation are low, the perceiver may utilize heuristics as the sufficiency threshold is lower. In contrast, if accuracy motivation and cognitive resources are high, the perceiver will engage in systematic processing. The perceiver raises the sufficiency threshold to a higher point to feel adequately confident that the assessment satisfies the accuracy concerns.

3. Heuristic-systematic model and code

Although the HSM was developed for constructs in the social psychology area, it can be used to describe trust in code and code reuse. Programmers are taught various conventions in their training such as making code easy to read, organized, and appropriately commented (Gaddis, 2016). These conventions are cognitive heuristics that are engaged when examining code. Code that is referred to as “spaghetti code” does not adhere to these conventions, either from poor initial writing or from being rewritten by several programmers, which explains why “spaghetti code” is often abandoned and not reused. We contend that programmers use cognitive heuristics to determine their initial trustworthiness of code.

A recent cognitive task analysis (CTA; Alarcon et al., 2017) found three factors that influence trust in code and code reuse: reputation, transparency, and performance. Reputation was defined as trustworthiness cues based on meta-information associated with the code, such as source, number of reviews, and number of users of the code, to name a few. Transparency was defined as the perceived comprehensiveness of the code from viewing it. Lastly, performance was defined as the capability of the code to meet the requirements of the project (Alarcon et al., 2017). From these definitions, it is apparent that reputation and transparency are largely made up of norms and conventions accepted by computer science and information technology professionals. Reputation is largely assessed outside of the code, from sources and reviews of the code. Alarcon and Ryan (in press) note the similarity of reputation to the credibility research in the persuasion literature. Transparency refers to following conventions from the computer science literature, which can also lead to heuristic processing when reading and evaluating source code text. In contrast, performance is related to systematic processing as the programmer has to analyze and check the code in-depth. We focus on initial viewing of the code with the reputation and transparency constructs.

The reputation and transparency factors are comprised of several facets (Alarcon et al., 2017). To make the study manageable, we focused on a single, simplified aspect of reputation, and two complex aspects of transparency. Reputation is meta-information a programmer acquires regarding the code, such as the source or origin of the code (e.g. library, coworker), reviews of the code (e.g. average rating of the code), contributors, or users (e.g. how many downloads over a period of time). Reputation can be assessed from multiple sources, such as exploring a trusted library for code with good reviews and
a large number of recent downloads. It is important to note that reputation of the code does not have to entail viewing the code, and as such is mainly a heuristic process. Reputation is similar to credibility in the persuasion literature (Alarcon & Ryan, in press), which has been found to have a strong influence as a heuristic (Chaiken & Maheswaran, 1994). For the current studies discussed, we focused only on the source and whether it was unknown or reputable, neither of which were further qualified.

Transparency is comprised of several facets in the CTA by Alarcon et al. (2017). Two aspects of particular importance are readability and organization. Readability and organization are often introduced as best practices (Gaddis, 2016). These best practices are heuristics as they are norms set forth by the industry, but do not necessarily need to be followed to write functioning code. Experienced programmers often encounter code that is not organized or easily readable. This may be due to expediency in producing the code, poor habits on the part of the programmer, or naïve programming skills. Readability is the ease with which a programmer can review and understand the intended behavior of the code. Readability is key to transparency because programmers must be able to easily ascertain the intent of the described functionality when deciding whether to reuse the code. Often, this can be determined from a quick review of the code (e.g., misuse of case or placement of braces and improper line length). Albayrak and Davenport (2010) specifically examined the effects of indentation and naming defects on code inspection, which target readability. Results showed higher false positives in finding perceived defects when inspecting degraded code, indicating distrust. Organization relates to the control structure and orientation of the code description. Highly organized code is represented logically and has an expected flow. The proper placement of variables, methods, and functions is easily detected at first glance and should initiate heuristic processing. The authors could not find any research to date on examining the organization of code and its impact on trust.

Heuristic processing has demonstrated an effect on perceptions of trust in other fields of psychology. Trust can be viewed as a heuristic itself (Tortosa-Edo, López-Navarro, Llorens-Monzonís, & Rodríguez-Artola, 2014), but the current study is concerned with the formation of the trust heuristic and the processes that lead to trust. Heuristics such as type of communication (video vs. text; Kim & Sundar, 2016), content readability (Agnihotri & Bhattacharya, 2016), information quality (Yi, Yoon, Davis, & Lee, 2013), reputation of a system (Zhou, 2012), and quality of a system (Zhou, 2012) all have demonstrated an influence on subjective trust assessments in some form. If code is from a reputable source, is easily read, and is well organized, trust in code should be higher due to matching heuristics across multiple conventions found in the computer programming literature.

Hypothesis 1: Code indicated as coming from a reputable source will lead to increased trust beliefs.

Hypothesis 2: More readable code will lead to increased trust beliefs.

Hypothesis 3: More organized code will lead to increased trust beliefs.

One of the key tenets of the HSM is that heuristic processing takes less time and cognitive effort (Chen et al., 1999). Chaiken (1980) found participants with greater involvement and higher regards for accuracy took more time to read persuasive arguments compared to those with less involvement and concern. She concluded systematic processing had occurred because more time was taken in the former than the latter. Additional research has generally supported the notion that when processing time is limited, heuristic processing occurs, and when time is abundant, systematic processing occurs. For example, when a “Take the Stairs” sign was presented near stairs (point of choice is near and processing time is limited), participants took the stairs more than the escalator located further away (Suri, Sheppes, Leslie, & Gross, 2014). In contrast, signs that presented systematic processing, such as “Will you take the stairs?” had stronger effect when further away from the point of choice. We contend source, readability, and organization are heuristic processes related to significantly less time assessing code as the heuristic processes adhere to the conventions set forth in the computer science industry.
Hypothesis 4: Code indicated as coming from a reputable source will lead to less time assessing the code.

Hypothesis 5: More readable code will lead to less time assessing the code.

Hypothesis 6: More organized code will lead to less time assessing the code.

4. Aim of current research
The current paper explores the relationship of the reputation and transparency factors found by Alarcon et al. (2017) on perceptions of code trustworthiness and time spent reviewing the code. Although Alarcon et al. (2017) hypothesized the factors should influence trust perceptions, little empirical research to date has tested the effects of the constructs on their criterion. Therefore, the current paper explored the effects of source, readability, and organization of code on trustworthiness assessments and time spent on code. In order to replicate the findings, a second experiment was conducted on a convenience sample of participants through an online crowd sourcing website using the same code, code manipulations, and order of manipulations as the first experiment. The authors have posted the stimuli and order in the supplemental materials. More importantly, we conducted the same experiment with the same stimuli both in a laboratory setting and online to test our hypotheses.

5. General method

5.1. Overview
In the studies reported below, we used images of computer code and manipulated the reputation and transparency of 18 pieces of computer code. The purpose of these studies was to investigate how the differences in the manipulations influenced coder’s trustworthiness ratings, remarks about the code, and time spent reviewing each piece. The experiment was administered to two separate samples. The first study was conducted in controlled settings and the experimenter was present. The second study sought to replicate the findings of study 1 using Amazon Mechanical Turk (MTurk). Because this type of research had never been empirically tested previously, we wanted to ensure our findings were not the result of chance. All stimuli and instructions were the same in both studies.

5.2. Stimuli
All stimuli were Java classes. Java was chosen because it is one of the most popular programming languages. Readability and organization guidelines were derived from Java Style Guides (Gaddis, 2016; Google, 2015; Sun Microsystems, 1997), a search of questions and answers on stackoverflow.com, and a textbook for coding standards (Gaddis, 2016). Code was downloaded from public repositories at github.com and degraded according to the guidelines mentioned above. Readability degradations consisted of (1) misuse of case, (2) misuse of braces, (3) misuse of indentation, and (4) improper line length and line wrapping, see Table 1. Organization degradations consisted of (1) poor grouping of methods, (2) misuse of declarations, (3) ambiguous control flow, (4) improper exception handling, and (5) statements unnecessarily requiring additional review, see Table 2. Reputation manipulations consisted of a heading above the code indicating whether it came from a “Reputable” or “Unknown” source. Participants were free to review the entirety of the presented code to assess its quality; we included these source headings to test whether minimal information about the origin of the code would have a significant influence on their assessments. Reputation manipulations attempted to emulate computer programming norms of “trustworthy” or “unknown” websites. The term “trustworthy” was not used to label the source of the code, as it may unduly bias their responses on the trustworthiness scale. Participants were to make trustworthiness assessments themselves. The participants were able to insert their own heuristic of a reputable source when making the assessment. In addition, if we provided a source we deemed as reputable, the partici-
pant may not have viewed the source the same way, adding additional error variance. Figure 1 illustrates a sample piece of code, the trustworthiness scale, and the use/don't use radio buttons. All comments were removed from of the code because programmers could ignore the actual code functionality in favor of what a comment claims the code could do. In addition, we did not want to bias the stimuli if a comment’s style or validity, unexpected by the programmer, was responsible for the distrust.

| Table 1. Readability degradations |
|-----------------------------------|
| 1. Misuse of case                  |
| (a) For packages                  |
| (b) For classes and interfaces    |
| (c) For methods and variables     |
| (d) For constants                 |
| 2. Misuse of braces               |
| (a) Line break before an opening brace |
| (b) No line break after an opening brace |
| (c) No line break before a closing brace |
| (d) Line break after a brace that precedes an else |
| (e) Missing a space before an opening or closing brace |
| 3. Misuse of indentation          |
| (a) Improper indentation given code position |
| (b) Inconsistent indentation      |
| 4. Improper line length and line wrapping |
| (a) Unnecessarily exceeds character limit without wrapping |
| (b) Missing blank lines to indicate logical grouping |
| (c) Use of too many and unnecessary blank lines |

| Table 2. Organization degradations |
|-----------------------------------|
| 1. Poor grouping of methods       |
| (a) Any form                      |
| 2. Misuse of declarations         |
| (a) Import statements used improperly |
| (b) More than one variable per line |
| (c) Variables not initialized as soon as possible |
| (d) Overuse of public instance and class variables |
| 3. Ambiguous control flow         |
| (a) Improper, unnecessary, or confusing use of “break” or “continue” |
| (b) Unnecessary or confusing nesting of blocks |
| (c) Multiple function calls or unnecessarily grouping block on one line |
| (d) Switch statement does not have a default case |
| (e) Switch statement with no “break” does not comment explicit continuation to next statement group |
| 4. Improper exception handling    |
| (a) Any form                      |
| 5. Statements unnecessarily require additional review |
| (a) Compressed if statements      |
| (b) Unusual return statements     |
| (c) Multiple classes              |
| (d) Inconsistent blocks           |
5.3. Materials

5.3.1. Background Surveys
Basic demographic information was collected using the Demographics Questionnaire (Chen, Barnes, Qu, & Snyder, 2010). Participants were also asked to indicate their primary and secondary programming languages, years of total programming experience, and years in current job position.

5.3.2. Trustworthiness
We used a single-item measure of overall trustworthiness. Participants indicated their trustworthiness of the code with the item “How trustworthy do you find this code?” on a Likert scale of 1 to 7, with 1 indicating “Untrustworthy” and 7 indicating “Trustworthy”. Single-item measures are appropriate when multiple item measures are likely to cause response fatigue and when the item is not ambiguous (Wanous, Reichers, & Hudy, 1997).

5.3.3. Remarks
Participants were provided with a section to make remarks on the code if they chose not to use the code. These remark sections were provided to verify that the manipulations were achieving the desired effect.

5.3.4. Time
The time spent evaluating the code was derived from the Hyper Text Markup Language (HTML) timestamps. A timestamp was used to indicate when a participant started a new page. The time between new page starts provided the total time spent on one page. Each page contained only one stimulus.

5.4. Procedure
Participants logged onto a website and filled out background surveys. After completion of the background surveys, the participants were informed they would be reviewing 18 pieces of code, all of
which compiled, and in which all comments were purposefully removed from the code. Their task was to assess the trustworthiness of the code and whether they would use the code or not for each of the 18 pieces of code. The code was manipulated according to the criteria described in the Stimuli section. Self-reports of perceived code trustworthiness and time spent on the code were the criteria of the experiment. Upon completion of the study, participants were debriefed and thanked for their time.

5.5. Data analysis
We used the Generalized Estimating Equation (GEE) for analyses. However, no good power analyses exist for determining sample size. As such, we conducted an a priori power analyses for repeated measures analysis of variance using G*Power (Faul, Erdfelder, Lang, & Buchner, 2007) with power \( (1 - \beta) \) set at 0.95 and \( \alpha = 0.05 \), two tailed to estimate our sample size. We estimated a medium effect size \( (ES = 0.20) \), as there is no previous literature on the topic. Correlation among the repeated measures was set to a small effect size \( (ES = 0.10) \) as differences in each measurement should be a result of the artifact stimuli presented as Java code, specifically a single Java class. Results from the power analysis indicated we needed a sample size of 38 to find our effect. However, recent research has indicated power issues in multiway ANOVAs due to hidden multiplicity (Cramer et al., 2016). As such, we oversampled, collecting approximately twice the size of the required sample size to ensure there was enough power to detect all significant main and interaction effects after discarding bad data.

We used a GEE approach to analyze data for trustworthiness and time. Repeated measures (RM) were incorporated into the study design for trustworthiness and time assessments. GEE analysis is useful for controlling for nuisance parameters, such as repeated measures, by estimating population average effects and using robust standard errors (Zeger, Liang, & Albert, 1988). Traditional RM ANOVA was not used for analysis for several reasons. First, it was not assumed the correlations between measures would have a stable correlation coefficient. RM ANOVA is typically used for longitudinal analyses, where measures over time are correlated with each other at some level. Expectations in the current study are that manipulations to the code will drive differences in measurement, and a single correlation coefficient does not adequately represent the data. Second, GEE allows one to specify a correlation structure. We opted for an unstructured correlation matrix as it is likely each correlation between pieces of computer code follows no meaningful pattern, including the time analysis. While an autoregressive correlation structure seems intuitive, participants may naturally spend less time on some pieces of code and more time on others, so a consistent autoregressive pattern is not expected between pieces of code. Third, RM ANOVA assumes normally distributed continuous outcomes; GEE models do not have such assumptions.

We conducted the same analytical procedure one would for a repeated measures ANOVA on the GEE analyses. Specifically, first we ran an omnibus test with the full factorial model. We reported main effects and interactions effects. If any three-way interaction effects were present, we ran subsequent simple comparisons with two-way interactions by source grouping. Simple effects were determined by plotting the means and standard errors of two-way interactions by grouping.

6. Study 1
This first experiment was conducted in-person, in controlled settings. Using an online website created for the purpose of this research, participants were asked to evaluate the trustworthiness of 18 pieces of computer code and whether they would use the code or not.

6.1. Participants
A total of 70 professional programmers from several aspects of industry were recruited to participate in the study in exchange for $50 (USD) financial remuneration. Participation required a minimum of three years of experience with software development and programmers had to know the Java language. Three participant’s data were not used due to server malfunctions, and any participant who listed “student” as their occupation was discarded. The data cleaning resulted in a total
sample of 53 usable participants. The sample was primarily male (96%) with a mean age of 35 years (range 20 to 60), a mean of 6 years in their current occupational position, a mean of 13 years of total coding experience, and 75% of participants listed Java as their primary programming language.

6.2. Procedure
The study used an experimental design with a convenience sample of programmers from the local area. Participants were greeted by the experimenter and told they would be examining images of code containing Java classes. After providing written consent to continue, participants were logged onto a website, filled out background surveys, and reviewed the 18 pieces of code. The website was accessed using either a desktop or laptop computer. Upon completion of the study, participants were debriefed, provided compensation, and then dismissed.

6.3. Results
6.3.1. Remarks
To determine if the manipulations to the code were being perceived by the participants appropriately, we analyzed the remarks sections of the code when participants chose not to use the code. After qualitative coding, the number of participants mentioning the manipulations coincided well with the condition they were found in. For medium readability, 21 participants made note of our readability degradations, and 56 commented on our degradations in the low readability conditions. Likewise, 24 participants highlighted our organization manipulations in the medium organization condition and 27 in the low organization condition. Finally, 46 participants commented that when a given piece of code was designated as coming from an unknown source, they chose “don’t use.” The qualitative data from the participants suggest that our attempt to manipulate the three factors in an ordinal fashion was successful. Correlation analyses indicated trustworthiness was negatively related to time spent reviewing code, $r = -0.26$, $p < 0.001$.

6.3.2. Trustworthiness
The GEE analyses indicated a statistically significant main effect of source on trustworthiness, Wald $\chi^2(1, N = 954) = 18.65$, $p < 0.001$, such that code that was from a reputable source ($M = 4.59$, SE = 0.13) was viewed as more trustworthy than code from an unknown source ($M = 4.08$, SE = 0.14), $z = 4.32$, $p < 0.001$. Hypothesis 1 was supported. A statistically significant main effect of readability was also observed, Wald $\chi^2(2, N = 954) = 40.67$, $p < 0.001$, such that high readability ($M = 4.85$, SE = 0.16) was viewed as more trustworthy than medium ($M = 4.05$, SE = 0.13), $z = 6.15$, $p < 0.001$, or low readability ($M = 4.11$, SE = 0.14) code, $z = 5.30$, $p < 0.001$. However, no significant differences were found between medium and low readability, $z = 0.54$, $p = 0.858$. Hypothesis 2 was supported. A main effect of organization was observed, Wald $\chi^2(2, N = 954) = 29.45$, $p < 0.001$, such that high organization ($M = 4.02$, SE = 0.14) was viewed as significantly less trustworthy than code that was medium ($M = 4.49$, SE = 0.13), $z = -4.74$, $p < 0.001$, or low in organization ($M = 4.50$, SE = 0.14), $z = -4.49$, $p < 0.001$. No significant differences were found between medium and low organization, $z = 0.12$, $p = 0.993$. Hypothesis 3 was not supported.

For the analysis of trustworthiness ratings, the three-way interaction between readability, organization, and source was significant, $\chi^2(4, N = 954) = 11.19$, $p = 0.025$. Two models were then constructed, breaking the data down between levels of source. There was a significant two-way interaction between readability and organization when source was reputable, $\chi^2(4, N = 477) = 9.96$, $p = 0.041$. Results are illustrated in Figure 2. Table 3 illustrates all means and standard errors. In order to understand the simple main effects of readability, three additional models were constructed for each level of organization. When organization was medium and source was reputable, readability had a significant simple main effect on trustworthiness, $\chi^2(2, N = 159) = 18.20$, $p < 0.001$, such that high readability had a mean of 5.25 (SE = 0.20), medium readability had a mean of 4.92 (SE = 0.23), and low readability had a mean of 4.09 (SE = 0.22). Pairwise comparisons suggested a mean difference existed between high and low, $z = 4.09$, $p < 0.001$, and low and medium, $z = -3.34$, $p = 0.002$, but not between high and medium, $z = 1.30$, $p = 0.396$. Readability also had a statistically significant effect on
Figure 2. Trustworthiness assessment plots of readability and organization by source for study 1.

Table 3. Study 1 and 2 trustworthiness scores and time means and standard errors

|                    | Unknown source |                      | Reputable source |                      |
|--------------------|----------------|----------------------|-------------------|----------------------|
|                    | Low            | Medium               | High              | Low                  | Medium               | High               |
| Low organization   |                |                      |                   |                      |                      |                   |
|                     | 4.189 (0.214)  | 2.887 (0.201)        | 4.868 (0.230)     | 4.887 (0.227)        | 4.642 (0.222)        | 5.547 (0.197)     |
|                     | 160.952 (13.278) | 205.761 (19.908)    | 280.629 (31.978)  | 183.245 (15.098)     | 263.488 (28.936)     | 156.244 (13.595)  |
| Medium organization | 3.811 (0.240)  | 4.358 (0.214)        | 4.509 (0.243)     | 4.094 (0.216)        | 4.925 (0.232)        | 5.245 (0.201)     |
|                     | 197.715 (20.536) | 150.229 (12.902)    | 123.268 (7.988)   | 197.570 (16.015)     | 166.291 (18.413)     | 180.451 (11.678)  |
| High organization  | 3.717 (0.219)  | 3.623 (0.239)        | 4.792 (0.233)     | 3.962 (0.242)        | 3.868 (0.210)        | 4.151 (0.271)     |
|                     | 225.355 (25.725) | 116.916 (11.664)    | 117.987 (9.485)   | 276.441 (26.313)     | 240.180 (26.548)     | 166.715 (13.283)  |
| Study 2             |                |                      |                   |                      |                      |                   |
| Low organization   | 4.507 (0.143)  | 4.370 (0.165)        | 5.397 (0.122)     | 5.219 (0.155)        | 5.219 (0.147)        | 5.260 (0.173)     |
|                     | 90.154 (11.518) | 113.241 (20.004)    | 128.767 (18.365)  | 84.533 (13.272)      | 96.863 (11.650)      | 107.185 (20.796)  |
| Medium organization | 4.69 (0.178)   | 4.411 (0.169)        | 4.849 (0.194)     | 5.027 (0.149)        | 5.466 (0.124)        | 5.151 (0.152)     |
|                     | 115.136 (30.795) | 97.589 (20.053)     | 67.106 (9.533)    | 94.857 (11.928)      | 81.079 (12.614)      | 73.118 (7.679)    |
| High organization  | 4.151 (0.177)  | 4.507 (0.178)        | 4.548 (0.155)     | 4.932 (0.213)        | 4.726 (0.164)        | 4.699 (0.193)     |
|                     | 109.205 (16.439) | 129.008 (29.133)    | 56.424 (7.762)    | 152.667 (59.755)     | 78.660 (10.605)      | 77.014 (12.812)   |

Note: Bold numbers are trustworthiness scores, non-bold numbers are time in seconds, and standard errors are in parentheses.
mean trustworthiness when organization was low and source was reputable, \( \chi^2(2, N = 159) = 14.50, p < 0.001 \), in which the estimated means were 5.55 (SE = 0.20) for high, 4.64 (SE = 0.22) for medium, and 4.89 (SE = 0.23) for low readability. Pairwise comparisons found significant mean differences between high and low, \( z = 2.44, p = 0.039 \), and high and medium, \( z = 3.68, p < 0.001 \), but not for low and medium, \( z = 0.88, p = 0.655 \). Readability did not have a significant effect on trustworthiness when organization was high and source was reputable, \( \chi^2(2, N = 159) = 0.80, p = 0.671 \).

In the second set of models created for unknown source, the two-way interaction between readability and organization was significant, \( \chi^2(4, N = 477) = 29.27, p < 0.001 \). The simple main effect of readability was statistically significant, \( \chi^2(2, N = 159) = 27.85, p < 0.001 \), when organization was high and source was unknown. The marginal means were estimated to be 4.79 (SE = 0.23) for high readability, 3.62 (SE = 0.24) for medium, and 3.72 (SE = 0.22) for low. The marginal mean for high readability was statistically different from both low, \( z = 3.71, p < 0.001 \), and medium, \( z = 5.11, p < 0.001 \); low and medium readability were not statistically significant, \( z = 0.36, p = 0.932 \). Readability continued to have a significant effect on trustworthiness when organization was low and source was unknown, \( \chi^2(2, N = 159) = 47.88, p < 0.001 \). High readability had a mean of 4.87 (SE = 0.23), medium had a mean of 2.89 (SE = 0.20), and low had a mean of 4.19 (SE = 0.21). Pairwise comparisons found significant differences between the means of high and low, \( z = 2.75, p = 0.017 \), and high and medium, \( z = 6.03, p < 0.001 \), and low and medium, \( z = 5.54, p < 0.001 \). The effect of readability was not statistically significant when organization was medium and source was unknown, \( \chi^2(2, N = 159) = 5.51, p = 0.064 \).

### 6.3.3. Time

The purpose of analyzing the time it took a participant to assess a piece of code was to highlight possible instances in the review process that required more or less cognitive effort, where longer times indicate greater effort. However, time was confounded by the fact that participants were prompted to write about why they would not use a given piece of source code. There were also potential differences between individuals in regard to how much they wrote in a given condition. To control for this, the number of words a participant wrote was calculated for each participant in each condition. Word counts were then included as a covariate to control for the amount of time one took writing in our GEE analyses of time. The average number of words written by a single participant in a single condition overall was 7.42. The average number of words written across readability levels for high, medium, and low was 4.51, 8.75, and 9.00, respectively. The average number of words across organization levels was 9.07, 6.96, and 6.22 for high, medium, and low, respectively. Finally, conditions with code from a reputable source averaged 7.24 words and unknown source averaged 7.60 words.

The GEE analyses indicated a significant effect of word counts on time, Wald \( \chi^2(1, N = 954) = 37.96, p < 0.001 \). All estimated marginal means have the intercepts of the model set at 7.42 words. The average word count. All time estimates are in seconds the GEE analyses indicated a main effect of source on time spent on code, Wald \( \chi^2(1, N = 954) = 9.29, p = 0.002 \), such that participants spent more time on code from a reputable source (\( M = 203.40, SE = 12.41 \)) than code from an unknown source (\( M = 175.42, SE = 11.31 \)). Hypothesis 4 was not supported. A statistically significant main effect of readability was also observed, Wald \( \chi^2(2, N = 954) = 25.26, p < 0.001 \), such that participants spent less time on code that was high in readability (\( M = 170.88, SE = 9.78 \)) than medium (\( M = 190.47, SE = 14.02 \)) or low readability code (\( M = 206.88, SE = 11.96 \)), but low was not significantly different from medium readability. Hypothesis 5 was supported. A main effect of organization was also observed, Wald \( \chi^2(2, N = 954) = 25.41, p < 0.001 \), such that participants spent more time on code that was high in organization (\( M = 190.60, SE = 11.35 \)) than code medium (\( M = 169.25, SE = 10.95 \)) in organization. In addition, code medium in organization was significantly different than low organization (\( M = 208.39, SE = 14.05 \)). No significant differences were found between high and low organization code. Hypothesis 6 was partially supported.
The model analyzing length of time reviewing the code revealed a statistically significant model effect for the three-way interaction between readability, organization, and source when controlling for word counts, \( \chi^2(4, N = 954) = 32.00, p < 0.001 \). Means and standard errors are reported in Table 3 and illustrated in Figure 3. First, readability and organization had a significant two-way interaction when source was reputable, \( \chi^2(4, N = 477) = 28.58, p < 0.001 \). Holding organization constant at high and source at reputable, readability maintained a statistically significant effect on time, \( \chi^2(2, N = 159) = 17.54, p < 0.001 \). Low readability (\( M = 282.42, SE = 25.63 \)) and medium readability (\( M = 248.47, SE = 26.47 \)) had the highest times for assessment, though they were not significantly different from one another, \( z = 1.01, p = 0.569 \). High readability (\( M = 180.28, SE = 15.14 \)) had the lowest time and was significantly different from both low readability, \( z = −3.36, p = 0.002 \), and medium readability, \( z = −3.14, p = 0.004 \). When organization was degraded to low and source was reported as being reputable, readability had a significant simple main effect on time when controlling for word counts, \( \chi^2(2, N = 159) = 17.27, p < 0.001 \). High readability (\( M = 144.02, SE = 13.69 \)) was not significantly different from low readability (\( M = 171.24, SE = 14.90 \)), \( z = −1.68, p = 0.213 \). High readability was significantly different from medium (\( M = 251.83, SE = 30.20 \)), \( z = −3.91, p < 0.001 \), and low was significantly different from medium, \( z = −2.60, p = 0.025 \). Readability did not have a main effect on time when organization was medium and source was reputable, \( \chi^2(2, N = 159) = 3.75, p = 0.153 \).

Next, the simple main effect of readability on time was statistically significant, \( \chi^2(2,N = 159) = 23.13, p < 0.001 \), when organization was high and source was unknown. High readability had the lowest marginal time average, with a mean of 115.72 s (\( SE = 8.60 \)), followed by medium (\( M = 122.91, SE = 11.37 \)) and low (\( M = 233.88, SE = 25.33 \)). There were significant mean differences between high and low, \( z = −4.78, p < 0.001 \), and low and medium, \( z = −4.10, p < 0.001 \), but not for high and medium, \( z = −0.64, p = 0.798 \). As organization was degraded to medium and source was kept at unknown, readability continued to significantly affect the time it took to assess code, \( \chi^2(2, N = 159) = 18.43, p < 0.001 \). High readability took the least amount of time, with a marginal mean of 116.80 s.
(SE = 8.35), followed by medium (M = 144.05, SE = 12.15) and low (M = 197.74, SE = 20.70). High and low readability were significantly different, \( z = -4.29, p < 0.001 \), and low and medium were as well, \( z = 3.23, p = 0.004 \); high and medium readability did not differ significantly, \( z = -2.15, p = 0.080 \). Finally, readability had a significant effect on time when organization was held constant at low and source at unknown, \( \chi^2(2, N = 159) = 15.74, p < 0.001 \). In this case, high readability led to the longest average times (M = 280.22, SE = 31.80), followed by medium (M = 213.21, SE = 20.90) and low (M = 160.83, SE = 12.85). High and low readability significantly differed, \( z = 3.75, p < 0.001 \), as did low and medium readability, \( z = -2.45, p = 0.039 \); high and medium readability did not significantly differ in average times, \( z = 2.09, p = 0.092 \).

6.4. Discussion
Study 1 found that manipulations to source code according to the trustworthiness factors described in Alarcon et al. (2017) resulted in significant differences in trustworthiness assessments and time spent on code. In general, the effects were in the expected directions, with reputable source and higher readability leading to higher trustworthiness assessments. Interestingly, once readability was degraded, no differences were found between the medium and low conditions for both trust and time assessments. This indicates heuristic processing as code that meets conventions takes less time to process. In contrast, code that was viewed as less trustworthy and did not meet conventions indicated a transition to systematic processing as more time was spent on the code to determine its underlying problems. However, organization had the opposite effect on trustworthiness. Organization may involve systematic processing instead of heuristic processing. The HSM predicts that heuristics will influence decisions to systematically process (Maheswaran & Chaiken, 1991) and bias those later decisions (Chaiken & Maheswaran, 1994). If code was from a reputable source it may have led to more in-depth analysis, particularly if the code was unorganized. Readability then facilitated systematic processing enabling faster processing of the code. The increased time spent on the code led to familiarity and understanding of the code which led to higher trustworthiness assessments. In contrast, if code was from an unknown source and unorganized, it led to an increase in time reviewing the code, but only if the code was at least moderately readable. If the code was low in readability, less organized, and from an unknown source, less time was spent on review. The reputation of the code had a biasing effect in the processing of the code at the lower readability levels.

7. Study 2
Study 2 attempted to replicate Study 1 using Amazon’s MTurk. MTurk is an online crowdsourcing website where tasks or experimental research can be conducted via the online labor market provided by Amazon. Researchers in the past several years have utilized the online labor markets for research (Zhou & Fishbach, 2016). Requirements to participant were the same as Study 1. However, due to the nature of the collection method, consent was embedded into the online study before the participants could begin the survey. In order to maintain some continuity, participants were prohibited from completing the survey using mobile devices.

7.1. Participants
A total of 127 online participants claiming to be programmers were recruited for the study. Research has demonstrated MTurk samples typically underestimate their cleaning processes, as researchers do not state how many participants they did not give credit to, only the cleaning process after rejection (Zhou & Fishbach, 2016). To remedy this, we report all cleaning procedures. We excluded any participant that had no variance in their trustworthiness score, which would indicate not paying attention to the study. We also excluded any participant that indicated a profession of “student.” The remaining 73 MTurk programmers indicated they were from various sectors of industry. Participation requirements were stated the same as in Study 1. Participants were paid $10.00 USD for participation as opposed to $50.00 in Study 1, as $50.00 compensation is not typical on MTurk and could unduly influence participants to participate. The sample was primarily male (83%) with a mean age of 29 years (range 20–54) and mean years of total programming experience was 7.7 (range 3–32), with 41% listing Java as their primary programming language.
7.2. Procedure
Study 2 was posted to Amazon MTurk. Once participants clicked on the survey link, they were directed to our website, they were provided with a brief description of the survey, and gave consent. Then, participants filled out demographic questionnaires and reviewed the 18 pieces of computer code. Upon completion of the survey, participants were shown a final screen with the debrief message and thanked for their time. Our team provided compensation through MTurk to participants.

7.3. Results

7.3.1. Remarks
As with Study 1, we qualitatively coded the remarks sections according to the degradations. For medium readability, 34 participants noted our readability degradations, 78 commented on our degradations in the low readability condition. Similarly, 14 participants highlighted our degradations in the medium organization condition and 21 in the low organization condition. Lastly, 12 commented the code was from an unknown source. Trustworthiness was not significantly related to time spent on code, $r = -0.03, p = 0.22$.

7.3.2. Trustworthiness
The GEE analyses indicated a main effect of source on trustworthiness, Wald $\chi^2(1, N = 1314) = 36.00, p < 0.001$, such that code that was from a reputable source ($M = 5.08, SE = 0.06$), was viewed as more trustworthy than code from an unknown source ($M = 4.60, SE = 0.06$), $z = 6.07, p < 0.001$. Hypothesis 1 was supported. A main effect of readability was also observed, Wald $\chi^2(2, N = 1314) = 6.35, p = 0.04$, such that high readability ($M = 4.98, SE = 0.07$) was viewed as more trustworthy than low readability ($M = 4.76, SE = 0.07$), $z = 2.34, p = 0.050$, but no differences were found between high and medium ($M = 4.78, SE = 0.07$), $z = 2.14, p = 0.083$, or low and medium, $z = -0.29, p = 0.955$. Hypothesis 2 was supported. A main effect of organization was also found, Wald $\chi^2(2, N = 1314) = 18.27, p < 0.001$, such that high organization ($M = 4.59, SE = 0.07$) was viewed as significantly less trustworthy than code that was medium in organization ($M = 4.93, SE = 0.07$), $z = -3.42, p = 0.002$, or low in organization ($M = 5.00, SE = 0.06$), $z = -4.15, p < 0.001$. No differences were found between medium and low organization, $z = 0.68, p = 0.776$. Hypothesis 3 was not supported.

When analyzing trustworthiness ratings with a full factorial GEE model, a three-way interaction between readability, organization, and source was observed, Wald $\chi^2(4, 1314) = 9.94, p = 0.041$. Means and standard errors are reported in Table 3 and illustrated in Figure 4. This model was then broken down and re-run by levels of source. When source was reputable, there was not a significant two-way interaction between readability and organization, Wald $\chi^2(4, 657) = 3.53, p = 0.473$. There was, however, a significant two-way interaction between readability and organization when source was held constant at unknown, Wald $\chi^2(4, 657) = 15.74, p = 0.003$. The simple main effects were then estimated for readability by all levels of organization. When source was unknown and organization was low, readability significantly affected trustworthiness ratings, Wald $\chi^2(2, 219) = 29.42, p < 0.001$. High readability had the highest trustworthiness ratings, with an estimated marginal mean of 5.32 (SE = 0.12), followed by medium readability ($M = 4.49, SE = 0.15$), and finally low readability ($M = 4.46, SE = 0.13$). High readability did significantly differ from both low, $z = 4.88, p < 0.001$, and medium, $z = 4.18, p < 0.001$, but low and medium did not differ, $z = -0.16, p = 0.986$. Readability was neither a significant effect on trustworthiness when organization was medium and source was unknown, Wald $\chi^2(2, 219) = 1.15, p = 0.562$, nor when organization was medium and source was unknown, Wald $\chi^2(2, 219) = 3.15, p = 0.207$.

7.3.3. Time
As in Study 1, word counts were included as a covariate to control for the amount of time one took writing in our GEE analyses of time. The average number of words written overall was 2.49. The average number of words written across readability levels for high, medium, and low was 1.35, 2.73, and 3.39, respectively. The average number of words across organization levels was 2.88, 2.53, and 2.06 for high, medium and low, respectively. Finally, conditions with code from a reputable source averaged 2.04 words and unknown source averaged 2.94 words.
The GEE analyses indicated no main effects of source, Wald $\chi^2(1, N = 1314) = 0.34, p = 0.561$, readability, Wald $\chi^2(2, 1314) = 3.76, p = 0.153$, or organization, Wald $\chi^2(2, 1314) = 2.95, p = 0.229$, on time. Table 3 illustrates the means and standard errors. Word counts for the remarks were a significant predictor of variance, Wald $\chi^2(1, N = 1314) = 42.99, p < 0.001$. Hypotheses 4–6 were not supported. The GEE analyses revealed the three-way interaction was not statistically significant, Wald $\chi^2(4, N = 1314) = 4.66, p = 0.324$. The only interaction that significantly affected by review time was the two-way interaction between readability and organization, Wald $\chi^2(4, 73) = 11.81, p = 0.019$. Figure 5 illustrates the results of the interaction. After removing source from the equation, readability had a main effect on time when organization was high, Wald $\chi^2(2, 438) = 8.22, p = 0.016$. High readability had the lowest estimated average time ($M = 67.07, SE = 7.54$), followed by medium ($M = 105.31, SE = 15.50$), and then low ($M = 134.74, SE = 31.39$). The simple main effect was significant, the pairwise comparisons found no significant differences between high and low, $z = −2.08, p = 0.095$, high and medium, $z = −2.21, p = 0.069$, or low and medium, $z = 0.84, p = 0.678$. Readability did not maintain a significant effect on time when organization was held at medium, Wald $\chi^2(2, 438) = 4.70, p = 0.095$, or low, Wald $\chi^2(2, 438) = 3.98, p = 0.136$.

7.4. Discussion

The results of Study 2 largely replicated the results of Study 1, specifically for trustworthiness. Trustworthiness assessments were all in the same direction as Study 1. However, no main effects were found for any of the manipulations in the time analyses in Study 2. Several explanations for the differences in time exist. First, the online MTurk participants may not have taken the experiment as seriously as the in-person participants. Researchers have warned about using MTurk as participants may not respond thoughtfully. To counteract this, researchers have suggested having manipulation checks (Fleischer, Mead, & Huang, 2015) or adding language that explains the importance of honestly responding (Zhou & Fishbach, 2016). In addition, both time and word counts in the remarks were lower than Study 1. The same main effects of the manipulations were found for trustworthiness. Second, differences in network speed or participants doing something else during the task may
have also affected reaction times as participants were not in a laboratory setting. However, the pattern of interactions for trustworthiness largely looks the same across the two studies. Although participants in Study 2 generally trusted more, as can be seen by the higher trust assessments, the patterns of the interactions were generally the same between both studies. In addition, an interaction of organization and readability was found for the time analyses. The pattern of low organization taking more time than high or medium organization in the high readability condition was similar to the effects in Study 1 for unknown source. High organization taking more time than low organization in the low readability condition was the same as Study 1 in both reputable and unknown conditions.

Third, group differences may explain the differences in the time analyses. Study 2 was collected from the internet, and as such may have had larger variations in coders than Study 1. Study 1 was mainly military programmers and contractors to the military. Study 2 participants most likely were not all military contractors. As such, differences between the samples may have led to the time differences. This may indicate differences between the types of coders recruited in Study 1 vs. Study 2.

8. General discussion
The reuse of computer code is an important issue for the computer science industry, both from an organizational and practical perspective. The present research sought to examine the effects of reputation and transparency on the perception of computer code from an HSM perspective. The current studies are some of the first, to our knowledge, to experimentally manipulate the psychological factors that influence trustworthiness perceptions. We explored not only self-reports of trustworthiness but also captured behaviors with time. In general, the two processes of heuristic and systematic processing occur in how programmers review and judge code. The study was largely replicable across two diverse samples. In addition, the current studies also expand the HSM to the human–computer interface literature, demonstrating heuristic and systematic processing are occurring when perceiving computer code.
8.1. Source
Source influenced all variables in the current study. If code was from a reputable source, programmers viewed the code as more trustworthy. However, in contrast to our hypotheses, reputable source led to increased time in reviewing the code. In the context of programming, source acts as a cue to continue on to systematic processing. In programming, the sufficiency threshold for confidence necessitates systematic processing, as no programmer would reuse code without knowing what it does. Poor credibility of the source may indicate a need to abandon processing because there is no need to perform systematic processing. Indeed, participants noted that, “server setup scripts should come from a known (trusted) server code provider” and, “server setup is mission critical enough to not rely on an unknown setup” when viewing the server code from an unknown source. These comments indicated a quick abandoning of the code due to a heuristic based on programmer experience with code reuse. These findings are in contrast to the persuasion literature on credibility which finds people tend to try to use as few resources as possible for making a decision (Carpenter, 2014; Chaiken & Maheswaran, 1994; Eagly & Chaiken, 1993). However, the goal in the persuasion literature may not always be systematic processing as is the case with programming. Reviewing code may be a process of deciding whether to perform systematic processing. In other words, programmers rely on heuristics to determine if it is worth their time and effort to perform systematic processing or abandon it (Alarcon & Ryan, in press). A recent meta-analysis illustrated that if people are systematically processing they are more persuaded by strong arguments (Carpenter, 2014). In contrast, if there is little systematic processing, people are persuaded more by heuristics. With the unknown source code, less systematic processing took place and participants were guided more by heuristics. Heuristic processing such as credibility can bias later systematic processing of information (Chaiken & Maheswaran, 1994; Chen et al., 1999), which also appears to be occurring in the evaluation of code. The source heuristic interacted with later heuristics of readability conventions and systematic processing of organization. These interactions will be discussed in the context of readability and organization below.

8.2. Readability
In general, code that was more readable was viewed as more trustworthy and took less time to evaluate. This may have occurred for two reasons. First, it is possible that code that is easily readable takes less time to process, regardless of whether heuristic or systematic processing occurs. Research in online reviews has found that readability of the review is positively related to perceived helpfulness (Schlosser, 2011) and purchase intentions (Jiménez & Mendoza, 2013). The readability manipulations of the current study focused heavily on accepted conventions and practices for writing Java code. For example, misuse of braces could make interpreting the code’s intent more difficult, as the braces would not be placed according to conventions, making it potentially difficult to understand the expected functionality the braces are encapsulating without an in-depth review. Improper and inconsistent indentation, given code position, could lead programmers to assume that code is being executed by different methods or portions of the script. Too many or too few blank lines could indicate that prior code has been deleted, sloppiness, or multiple programmers involved with the code over time. Blank lines are often used by programmers to keep lines of code logically grouped and understood together on the screen. Readability may enable participants to further explore the code when other issues arise, such as a sense that organization of the code is not as expected, thus facilitating systematic processing if other degradations arise.

A second interpretation is that code high in readability is easily recognized as being written carefully by someone comfortable with conventions, allowing heuristic processing to occur. The heuristic processing interpretation is supported by no significant differences in main effects for trustworthiness or time outcomes once the code was degraded. Research in online reviews has demonstrated that after readability has reached an ideal point, readability as a heuristic diminishes (Agnihotri & Bhattacharya, 2016). In code, readability has a degradation point where once degraded, heuristic processing is more likely to occur, leading to lower trustworthiness and reuse. Readable code could demonstrate that a knowledgeable and experienced programmer wrote the code, leading to higher trust inferences through a heuristic. The heuristic, in turn, facilitates systematic processing. Poorer readability may indicate a novice or disorganized team wrote the code, and as such a further examination is
necessary. This type of additive processing, where heuristic processing facilitates or inhibits systematic processing, has been demonstrated previously in the literature (Maheswaran & Chaiken, 1991).

8.3. Organization

Highly organized code led to significantly lower trustworthiness assessments and lower instances of use. Although these results seem counterintuitive, the HSM can assist in understanding the findings. Understanding code functionality and intent, organization of the internal code involves a systematic process, in contrast to readability which is easily assessed visually. Initial views of the code are heuristic processing, such as readability and source. This heuristic processing leads to a quick, possibly inaccurate assessment of the code. However, once participants decided to review the code in more detail, performing systematic processing, code high in organization took longer than code medium in organization, but not longer than code low in organization. The effects of organization must be interpreted with the interactions. When code was highly organized, participants spent more time on the code as readability was degraded. One interpretation is that it would be suspicious to an experienced programmer to review code with good organization but not follow readability conventions. In contrast, when code was low in organization, as readability decreased so did the time spent on the code. The organization degradations in the current studies focused on the logical and control structures applied to the writing and organization of the code in the program that would still allow the code to compile and function properly. The degradations disrupted the accepted practices associated with how the code for a Java class and its methods should appear and flow, leading to further investigation as seen through the time analyses. The organization degradations may have been perceived as occurring because the code was revised multiple times by different programmers with potentially different levels of skill. Degrading the organization of the code led to increased systematic processing as the programmers had to investigate the code to determine if the evident organization issues did not impact the functional intent of the code. This determination process also depended on readability of the code.

Poor organization likely facilitated increased systematic processing. Participants may have explored the code more due to the inconsistencies in the organization and readability, thus transferring from a heuristic processing approach (source and readability cues) to a systematic processing approach. This further assessment of the code encompassed more time with the code, enabling a deeper knowledge of the code and increasing trust perceptions. In addition, organization degradations are less explicit in code artifacts than readability degradations, as most readability degradations are immediately visually apparent; most organization degradations take both an understanding of the code structure and a deliberate concern that the structure is correct given that the code behavior was assumed correctly. This explains why high organization consistently had similar scores across readability conditions in reputable source but the time to evaluate the code took longer as readability was degraded. As organization was degraded, more cognitive effort was needed to determine the trustworthiness of the code, especially in the lower readability conditions, since it is inconsistent with programmer expectations. This extends the heuristic as a biased hypothesis of the HSM. Previous heuristic processes have demonstrated an effect of biasing systematic processing, particularly when messages were ambiguous (Chaiken & Maheswaran, 1994). The ambiguity of the code due to low organization and low readability, in those conditions, led to a bias in the systematic processing. Another biasing effect was source.

Source also had a biasing effect on the time spent on the code, given a level of readability and organization in Study 1. Organization as a systematic process is also influenced by previous heuristic processes, namely source, in that participants spent more time evaluating code degraded in organization if it was from a credible source in Study 1. This explains why highly organized code from a reputable source was relatively stable across the readability conditions, with a mean around 4, which indicates average trustworthiness because there were no conflicting perceptions between high organization and reputable source. Participants relied on the heuristics of source because the code was highly organized and thus did not require a deeper systematic processing look at the code as indicated by the less time spent on that code. However, source did not contribute to a three-way
interaction on time in Study 2, nor did any of the variables have main effects on time. This may be due to environmental differences in accuracy motivation.

8.4. Environment
The main effect of all manipulations and the three-way interaction present in Study 1 for the time analyses were not present in Study 2. This may have occurred due to slight differences in the samples. As mentioned before, Study 1 consisted of mainly military programmers and programmers contracted through the military. These programmers may have different organizational constraints placed on them for their code reuse, and as such had different accuracy motives. Indeed, military operations often have higher scrutiny because of the risks and requirements needed for software products. These constraints lead to a difference in the accuracy motivation, raising the sufficiency threshold for programmers in Study 1. As mentioned previously, the sufficiency threshold is a combination of effort required and cognitive resource availability, but if there are different motivations the thresholds will be different. Study 2 participants may have been utilizing more heuristic processing as they did not have higher accuracy motivations either from (1) not being in a military context, (2) being studied through complete computer mediation, or (3) not wanting to invest the cognitive resources in determining accuracy because of lower pay. These environmental differences may have explained the differences in results between the two studies. Indeed, Alarcon et al. (2017) discussed the moderating effects of environment on code trustworthiness. Although we did not ask the background or application of their programming experience in Study 2, it is reasonable to believe the participants were not all from a military background, which may have accounted for some of the differences between the studies such as no main effects or three-way interaction on time, less time being spent on the code, overall higher trustworthiness assessments, and higher reuse.

8.5. Limitations
The current study is not without limitations. First, this paper explored programmers’ overall trust in samples of code and largely confirms best practices set by the computer science industry, but it is important to recognize there could be other factors that might influence trust in a code artifact. Comments within code can be especially useful because a well-commented but poorly written code artifact might incite more trust than a perfectly written but poorly commented code artifact. With comments deliberately removed from the code, the reviewers possibly had to rely on their own writing experience, rather than accepted practices that they might have forgotten or did not deploy, to determine the functionality the code was asserting. It might be a difference in a programmer trained as a computer scientist vs. a programmer trained via another engineering discipline or self-taught.

Second, each piece of code might have its own unique trustworthiness level, despite the manipulations performed on the code. In one example, the code that was high in readability, high in organization, and with an unknown source was server code. Several participants noted they would not use server code from an unknown source. The high-security aspect of software for servers necessitates that code be from a reputable source. As such, it is unclear whether the interactions found are functions of the code manipulations to express the degradations or idiosyncrasies related to the code itself. Although it might be easy to use this logic to understand organization’s odd effects, it must be remembered that organization’s main effects were across all pieces of code, not just the interactions. Future research should introduce a repeatable process that more thoroughly vets prospective code to ensure a consistent trustworthiness level prior to any manipulations.

Third, the studies examined the code as a series of images, factoring the practices of code review. Some participants indicated that they would have liked the ability to compile and execute the code to determine if met their testing standards. Such compilation and execution of the code were outside the scope of this study. Being able to analyze the code as it is running or feeding in inputs with known outputs could lead to a different form of trust in the code regardless of the code review’s perceived organization or readability. However, the goal of the study was to explore how programmers initially perceive the code, and whether they would then download or run tests on the code to determine its full trustworthiness level.
Fourth, readability degradations appeared a total of 53 times across the 18 artifacts, whereas organization degradations only appeared 38 times. This could cause the organization degradations to be more difficult to notice when readability degradations are present, leading a programmer to distrust a code artifact without analyzing it enough to discover the organization issues within the code artifact. The readability aspects are visual and easily assessed from viewing the code. Organization degradations were harder to assess as they required more cognitive effort but the readability degradations may have hindered such processing.

Fifth, time was used to determine systematic and heuristic processing. Although time has been used frequently in dual process models to determine processing (see Chaiken, 1980), other methods may be more appropriate. Study 2 participants were collected online, and various aspects may have contributed to the non-significant effects of time. Analyzing the remarks participants made during the study may be a viable approach to determine processing. Unfortunately, participants were not required to leave remarks. We suggest future research informed by the HSM should utilize timing and other behaviors as a confirmation of effortful processing for a given task.

8.6. Implications
Overall, both studies confirm that readability issues are more likely to elicit distrust in code during a review than organization issues. There are also indications that when programmers perceived an inconsistency across interacting degradations, such as low readability and high organization, suspicion that something hidden may be wrong leads to an increase in the time spent on review as well as a decrease in trust. Organization is a harder capability to master than readability. It is possible that programmers that perceived high organization expected high readability. That would mean that any level of degradation for readability would be more suspicious if the organization was high.

The oversimplification of source values to “unknown vs. reputable” may have caused too immediate of a perceived low trust in the code. Experienced programmers generally know the provenance of the code they are reviewing. Even if it is an online source with which they have no familiarity, they will investigate the source until they are comfortable with downloading and reviewing the code. As noted, this implied bias in the study may have affected the trustworthiness levels outside of any of the readability and organization degradations. Future studies should examine reputation alone and with finer granularity to determine its influence on code reuse.

The issue with timing not being consistent across the studies can be partially explained by how the studies were performed. In Study 1, the participants were in a room where they could be observed as they reviewed the code. As they were working professionals and taking time away from their jobs, they were not inclined to linger and at the same time they were generally committed to the effort. Study 2 participants were not observed because the study was conducted online by Amazon MTurk participants. Therefore, they could allow for interruptions during the study increasing the length of time or they could “power through” the review. Because they knew that there was only $10 at the end of the study, the longer they took the less the payout was worth.

Supplementary material
The supplementary material for this paper is available online at https://doi.org/10.1080/23311908.2017.1389640.

Funding
The authors received no direct funding for this research.

Competing Interests
The authors declare no competing interest.

Author details
Gene M. Alarcon1
E-mail: gene.alarcon.1@us.af.mil
Rose Gamble2
E-mail: gamble@utulsa.edu
Sarah A. Jessup1
E-mail: sarah.jessup.ctr@us.af.mil
Charles Walter2
E-mail: Charlie-walter@utulsa.edu
Tyler J. Ryan1
E-mail: tyler.ryan@csra.com
David W. Wood1
E-mail: david.wood.57.ctr@us.af.mil
Chris S. Calhoun2
E-mail: chris.calhoun@csra.com

1 Air Force Research Laboratory, Wright-Patterson AFB, OH, USA.
2 Tandy School of Computer Science, University of Tulsa, Tulsa, OK, USA.
3 CSRA Inc., Dayton, OH, USA.
Citation information
Cite this article as: Application of the heuristic-systematic model to computer code trustworthiness: The influence of reputation and transparency. Gene M. Alarcon, Rose Gambaia, Sarah A. Jessup, Charles Walter, Tyler J. Ryan, David W. Wood & Chris S. Calhoun, Cogent Psychology (2017), 4: 1389640

Note
1. All code is available in the supplemental material.

References
Agnihotri, A., & Bhattacharyya, S. (2016). Online review helpfulness: Role of qualitative factors. Psychology & Marketing, 33(11), 1006–1017. doi:10.1002/mar.20934
Alarcon, G. M., Milletto, L. G., Ryan, P., Jessup, S. A., Calhoun, C. S., & Lyons, J. B. (2017). A descriptive model of computer code trustworthiness. Journal of Cognitive Engineering and Decision Making, 11(2), 107–121. doi:10.1177/1555343416657236
Alarcon, G. M., & Ryan, T. J. (in press). Trustworthiness perceptions of computer code: A heuristic-systematic processing model. In Proceedings of the Hawaii International Conference on System Sciences, USA (Vol. 51, pp. 1–10).
Albayrak, O., & Davenport, D. (2010). Impact of maintainability defects on code inspections. In Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (Vol. 50, pp. 1–4). doi:10.1145/1852786.1852850
Allamanis, M., Barr, E. T., Bird, C., & Sutton, C. (2014). Learning natural code conventions. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, China (pp. 281–293). doi:10.1145/2635868.2635883
Carpenter, C. J. (2014). Making compliance seem more important: The “just-one-more” technique of gaining compliance. Communication Research Reports, 31(2), 163–170. doi:10.1080/08824096.2014.907144
Chaiken, S. (1980). Heuristic versus systematic information processing and the use of source versus message cues in persuasion. Journal of Personality and Social Psychology, 39(5), 752–766. https://doi.org/10.1037/0022-3514.39.5.752
Chaiken, S., Giner-Sorolla, R., & Chen, S. (1996). Beyond accuracy: Defense and impression motives in heuristic and systematic information processing. In P. M. Gollwitzer, J. A. Bargh, P. M. Gollwitzer, & J. A. Bargh (Eds.), The psychology of action: Linking cognition and motivation to behavior (pp. 553–578). New York, NY: Guilford Press.
Chaiken, S., Liberman, A., & Eagly, A. H. (1989). Heuristic and systematic information processing within and beyond the persuasion context. In J. S. Uleman, J. A. Bargh, J. S. Uleman, & J. A. Bargh (Eds.), Unintended thought (pp. 212–252), New York, NY: Guilford Press.
Chaiken, S., & Maheswaran, D. (1994). Heuristic processing can bias systematic processing: Effects of source credibility, argument ambiguity, and task importance on attitude judgment. Journal of Personality and Social Psychology, 66(3), 460–473. doi:10.1037/0022-3514.66.3.460
Chen, J. Y. C., Barnes, M. J., Qu, Z., & Snyder, M. G. (2010). Heuristic and systematic processing: Effects of source credibility, argument ambiguity, and task importance. In Proceedings of the 30th IEEE Working Conference on Source Code Analysis and Manipulation (pp. 7–9). Washington, DC: IEEE Computer Society.
Hautola, L. S. (2015, June 23). Programmers are copying security flaws into your software, researchers warn. CNET. Retrieved from http://www.cnet.com/news/programmers-are-copying-security-flaws-into-your-software-researchers-warn/
Hoff, K. A., & Bashir, M. (2015). Trust in automation integrating empirical evidence on factors that influence trust. Human Factors: The Journal of the Human Factors and Ergonomics Society, 57(3), 407–434. https://doi.org/10.1177/0018720815547570
Jiménez, F. R., & Mendoza, N. A. (2013). Too popular to ignore: The influence of online reviews on purchase intentions of search and experience products. Journal of the Academy of Marketing Science, 41(3), 392–414. doi:10.1007/s11747-013-0404-0
Jones, S. L., & Shah, P. P. (2016). Diagnosing the locus of trust: A temporal perspective for trustee, trustee, and dyadic influences on perceived trustworthiness. Journal of Applied Psychology, 101(3), 392–414. doi:10.1037/ apl0000041
Prevalence and remedies. Psychonomic Bulletin & Review, 23(5), 640–647. doi:10.3758/s13423-015-0911-5
CVE Mitre. (2014). Heartbleed OpenSSL bug [CVE-2014-0160]. Retrieved from https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-0160
Durumeric, Z., Kang, J., Adrian, D., Holdeman, A., Bailey, M., Li, F., & Paxson, V. (2015). The matter of Heartbleed. ICM, 14, 5–7. doi:10.1145/2663716.2663755
Eagly, A. H., & Chaiken, S. (1993). The psychology of attitudes. Orlando, FL: Harcourt Brace Jovanovich College Publishers.
Faul, F., Erdfelder, E., Lang, A.-G., & Buchner, A. (2007). G*Power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences. Behavior Research Methods, 39(2), 175–191. doi:10.3758/BF03193146
Fiske, S. T., & Taylor, S. E. (1991). Social cognition (2nd ed.). New York, NY: McGraw-Hill.
Flavivin, C., & Guinolli, M. (2006). Consumer trust, perceived security and privacy policy. Industrial Management & Data Systems, 106(5), 601–620. doi:10.1108/02635460610666403
Fleischer, A., Mead, A. D., & Huang, J. (2015). Inattentive responding in MTurk and other online samples. Industrial Organizational Psychology: Perspectives on Science and Practice, 8, 192–202. doi:10.1017/ios.2015.25
Gaddis, T. (2016). Starting out with programming logic and design (4th ed.). New York, NY: Pearson.
Gigerenzer, G., & Gaissmaier, W. (2011). Heuristic Decision Making. Annual Review of Psychology, 62(1), 451–482. doi:10.1146/annurev-psych-120709-143546
Google. (2015). Java Style Guidelines. Retrieved from https://google.github.io/styleguide/javaguide.html
Griffen, R. J., Neuwirth, K., Giese, J., & Dunwoody, S. (2002). Linking the heuristic-systematic model and depth of processing. Communication Research, 29(6), 705–732. doi:10.1177/00492430022237833
Grubb, B. (2014, April 15). Heartbleed disclosure timeline: Who knew what and when. Sydney Morning Herald. Retrieved from http://www.smh.com.au/it-pro/security-it/heartbleed-disclosure-timeline-who-knew-what-and-when-20140414-zq4rk
Harman, M. (2010). Why source code analysis and management are important. In Proceedings of the 10th IEEE Working Conference on Source Code Analysis and Manipulation (pp. 7–9). Washington, DC: IEEE Computer Society.
Harland, M. (2010). Why source code analysis and management are important. In Proceedings of the 10th IEEE Working Conference on Source Code Analysis and Manipulation (pp. 7–9). Washington, DC: IEEE Computer Society.
Heartbleed OpenSSL bug [CVE-2014-0160]. Retrieved from https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-0160
Chen, S., Duckworth, K., & Chaiken, S. (1999). Motivated heuristic and systematic processing. Psychological Inquiry, 10(1), 44–49. doi:10.1207/s15327965pi1001_6
Cramer, A. J., van Ravenzwaaij, D., Matzke, D., Steingroever, H., Wetzels, R., Grasman, R. P., ... Wagenmakers, E. (2016). Hidden multiplicity in exploratory multiway ANOVA:
Kim, K. J., & Sundar, S. S. (2016). Mobile persuasion: Can screen size and presentation mode make a difference to trust? *Human Communication Research, 42*(1), 45–70. doi:10.1111/hcre.12064

Maheswaran, D., & Chaiken, S. (1991). Promoting systematic processing in low-motivation settings: Effect of incongruent information on processing and judgment. *Journal of Personality and Social Psychology, 61*(1), 13–25. doi:10.1037/0022-3514.61.1.13

Mayer, R. C., Davis, J. H., & Schoorman, F. D. (1995). An integrative model of organizational trust. *Academy of Management Review, 20*, 709–734. doi:10.2307/2587892

Schlosser, A. E. (2011). Can including pros and cons increase the helpfulness and persuasiveness of online reviews? *The interactive effects of ratings and arguments. Journal of Consumer Psychology, 21*(3), 226–239. doi:10.1016/j.jcps.2011.04.002

Sun Microsystems. (1997). Java code conventions. Retrieved from http://www.oracle.com/technetwork/java/codeconventions-150003.pdf

Suri, G., Shennes, G., Leslie, S., & Gross, J. J. (2014). Stairs or escalator? Using theories of persuasion and motivation to facilitate healthy decision making. *Journal of Experimental Psychology: Applied, 20*, 295–302. doi:10.1037/xap0000026

Tortosa-Edo, V., López-Navarro, M. A., Llorens-Monzonís, J., & Rodríguez-Artola, R. M. (2014). The antecedent role of personal environmental values in the relationships among trust in companies, information processing and risk perception. *Journal of Risk Research, 17*(8), 1019–1035. doi:10.1080/13698323.2013.841726

de Visser, E., & Parasuraman, R. (2011). Adaptive aiding of human-robot teaming: Effects of imperfect automation on performance, trust, and workload. *Journal of Cognitive Engineering and Decision Making, 5*(2), 209–231. doi:10.1177/1555334X11410160

Wanous, J. P., Reichers, A. E., & Hudy, M. J. (1997). Overview job satisfaction: How good are single-item measures? *Journal of Applied Psychology, 82*(2), 247–252. doi:10.1037/0021-9010.82.2.247

Yi, M. Y., Yoon, J. J., Davis, J. M., & Lee, T. (2013). Untangling the antecedents of initial trust in Web-based health information: The roles of argument quality, source expertise, and user perceptions of information quality and risk. *Decision Support Systems, 55*(1), 284–295. doi:10.1016/j.dss.2013.01.029

Zeger, S. L., Liang, K. Y., & Albert, P. S. (1988). Models for longitudinal data: A generalized estimating equation approach. *Biometrics, 44*(4), 1049–1060. doi:10.2307/2531734

Zhou, H., & Fishbach, A. (2016). The pitfall of experimenting on the web: How unattended selective attrition leads to surprising (yet false) research conclusions. *Journal of Personality and Social Psychology, 111*(4), 493–504. doi:10.1037/pspa0000056