BJM BRITISH JOURNAL of MANAGEMENT
BRITISH ACADEMY OF MANAGEMENT

British Journal of Management, Vol. 0, 1–23 (2022)
DOI: 10.1111/1467-8551.12653

Rise of the Androids: The Reflection of Developers' Characteristics in Computerized Systems

Daphne Sobolev
School of Management, University College London, One Canada Square, London, E14 5AA, UK
Corresponding author email: d.sobolev@ucl.ac.uk

Due to their speed and accuracy, computerized decision-making and data-analysis systems are often perceived to be close to the ideal of unbounded rationality. Challenging this perception, this study explores the possibility that computerized systems reflect the individual characteristics of the developers who have designed and realized them. Through the analysis of interviews with high-frequency trading system developers and a survey of software developers working in diverse industries, it shows that developers' characteristics, which have been related to bounded rationality (e.g. experience and expertise), influence their codes' performance and errors. Computer codes also reflect the personal circumstances of the developers (e.g. deadlines and stress), and their values. However, whereas some developers value code characteristics that are congruent with the ideal of unbounded rationality, including speed and accuracy, others aim to achieve characteristics that could be incongruent with it (e.g. code readability and modularity). Developers' roles and organizational practices, such as testing procedures and code reviews, limit the expression of their characteristics in their codes. Nevertheless, developers can be often identified by reading the codes. Highlighting that developers transfer some of their characteristics to their codes, this study identifies elements of bounded rationality in decision-making systems and extends organizational decision-making research.

Introduction

Bounded rationality theories postulate that human decision-making processes are subject to biases that often result in suboptimal choices and performance (Koumakhov and Daoud, 2021; Simon, 1947/1997, p. 115). These biases have been attributed to a wide range of human characteristics, including experience (Croce, Ughetto and Cowling, 2020), expertise (Adams and Jiang, 2017), education (Anderson, 2013), personality traits, emotions (Delgado-García, De La Fuente-Sabaté and De Quevedo-Puente, 2010), limitations on the data volumes that people can process (Barber and Odean, 2008) and limitations on their processing speed (Kiss, Rodriguez-Lara and Rosa-Garcia, 2020). By contrast, computers are capable of processing greater amounts of data efficiently, accurately, and at ever-increasing speeds. Therefore, computerized decision-making and analysis systems have been perceived to be closer to the ideal of unbounded rationality (Lindebaum and den Hond, 2020). The capabilities of modern computers have led to their use in many industries, including finance (Chaboud *et al.*, 2014; MacKenzie, 2018a), healthcare (El-Bouri, Gue and Lip, 2021; Wang *et al.*, 2019), and business management (Hajli *et al.*, 2022; Nguyen and Malik, 2022). But are computerized systems indeed close to the ideal of bounded rationality? Or do they reflect their developers' characteristics, which have been associated with bounded rationality?

The objective of this study is to explore how developers' characteristics influence the codes (computer programs or software) that they write. Thus, it examines how developers' characteristics, which have been linked to human bounded rationality (e.g. experience, expertise and education), impact their code performance. Drawing on the literature about algorithmic biases (Kordzadeh and Ghasemaghaei, 2022), the relationship between developers' personality and performance in the software industry (Gomez and Acuña, 2014), and the recognition of individual differences (Orehek and Human, 2017), it asks which personal characteristics the developers transfer to their codes. In addition, it asks how organizational practices interact with this transference process, and what the results of this process are.

To answer these questions, this study qualitatively examines computerized system development in the high-frequency trading (HFT) industry (Study 1) and in a range of other financial, healthcare, and business management industries (Study 2). It focuses on HFT because HFT provides an extreme case for the investigation of the expression of human characteristics in codes. In HFT firms, practitioners develop decision-making codes that trade with stock exchange computers thousands of times faster than people do. For example, the time required for HFT codes to react to news ranges between microseconds and nanoseconds, whereas the blink of a human eye takes 400 ms (Manahov, 2016). Hence, once the codes are launched, developers can only monitor their performance or stop them. However, HFT codes trade without direct human intervention. In addition, HFT is highly influential. In fact, it accounts for more than 50% of the USA trading volume (Hoffmann, 2014; Zhou and Olivari, 2013). It significantly affects market efficiency (Conrad, Wahal and Xiang, 2015) and liquidity (Jarnecic and Snape, 2014).[1] Furthermore, it has high, persistent revenues (Baron *et al.*, 2019), and its returns are especially large in times of great volatility, such as during the COVID-19 pandemic (Patterson and Osipovich, 2020). As HFT comprises influential automated decision-making codes, it provides an edge case for the analysis of the transference of developers' characteristics. In addition to HFT, this paper examines code development in a range of industries in order to obtain insights about computerized systems in wider contexts. Study 1 draws on a series of interviews with HFT code developers, and Study 2 comprises a survey of developers in healthcare firms, management firms, and additional firms.

Overall, the results suggest that the individual characteristics of HFT code-developers, including experience, expertise, and education, influence their code performance and errors. A proportion of firms employ practices, such as coding rules, code reviews, and testing procedures, that restrict the expression of developers' individual characteristics in their systems. Nevertheless, developers' characteristics can often be identified by reading their codes. Furthermore, also in industries other than HFT, the individual characteristics of the developers are reflected in their codes, and this result holds when developers consider their roles in the development of the codes and their knowledge of the whole system. Moreover, the codes reflect the personal circumstances of the developers. Finally, developers value both code characteristics that are congruent with the ideal of unbounded rationality (e.g. efficiency, speed, and accuracy) and code characteristics that are incongruent with it (e.g. code readability, ease of maintenance, and elegance). Thus, they sometimes deliberately develop codes that are partially incongruent with the ideal of unbounded rationality.

This study contributes to the literature on the relationship between individuals' characteristics and organizational decision-making in three major ways. First, this study extends its scope to computerized decision-making. Previous research has contributed important insights into the way in which individuals' characteristics influence organizational decision-making (Delgado-García, De La Fuente-Sabaté and De Quevedo-Puente, 2010; Hillenbrand *et al.*, 2020) but has focused on organizational decision-making, which is carried out by people. Understanding the relationship between developers' characteristics and computerized decision-making is important because, in modern industries, a large proportion of decision-making is carried out by computers or based on the results of computerized data analysis.

Second, this study suggests that developers transfer some of their characteristics to decision-making systems, so that, for instance, those who are more experienced develop faster and more accurate codes than those who are less experienced, and those who are risk-seeking develop codes that take more risks than those who are risk-averse.

[1]Market efficiency refers to the extent to which price movements resemble a random walk (Conrad, Wahal and Xiang, 2015), whereas liquidity provision refers to the submission of orders at multiple prices to stock exchange books (Jarnecic and Snape, 2014).

Previous research has examined the effects of developers' characteristics on algorithmic biases (Fazelpour and Danks, 2021; Kordzadeh and Ghasemaghaei, 2022) and on system performance (Gomez and Acuña, 2014). However, it has been limited and separate. In particular, studies on algorithmic biases have focused on gender and racial discrimination, whereas studies on system performance have focused on the developers' big five personality traits. Both of these research streams have disregarded important developers' characteristics such as experience and expertise. Taking a qualitative approach, this study suggests that these factors, as well as additional characteristics that have not been studied before in organizational contexts (e.g. developers' code-related values and error-handling skills), influence the codes and hence organizational decision-making.

Third, this study identifies organizational practices that limit the extent to which developers' individual characteristics are expressed in decision-making software. Research has examined the impact of organizational practices and characteristics on performance (Bozec, Dia and Bozec, 2010; Eisenbeiß and Boerner, 2013; Kellard *et al.*, 2017) but has not investigated how organizational practices influence the expression of developers' characteristics in their codes. Establishing that firm practices, including testing procedures, code reviews, and job rotations encourage code homogeneity, this study highlights that certain firm procedures constrain the expression of individual differences between developers. Hence, it identifies boundary conditions of the effects of developers' characteristics on their codes.

The results of this study have applications for managers, the media, and the public. In particular, they could be used to improve hiring decision processes and training programs in firms that aim to produce efficient and accurate decision-making codes (e.g. financial, healthcare, and business management firms). Specifically, they highlight that system failures could be the result of the employment of developers who have an inadequate attitude towards error-handling. Failures of decision-making systems can negatively impact individuals, firms, and markets. For example, the flash crashes attributed to computerized trading systems erase billions of dollars (Galouchko, Torsoli and Ekblom, 2022; Kirilenko *et al.*, 2017). In addition, the results provide the media and the public with information about the codes' nature. The unprecedented reliance on computerized systems has sparked a debate about their use. For instance, in the context of HFT, media articles have asked 'men or machines: who runs the markets?' (Moore, 2013), warned that 'the rise of the machines leaves markets exposed' (Kawa, 2018), and concluded that 'machines are driving Wall Street's wild ride, not humans' (Isidore, 2019). This is especially disconcerting to many, given that HFT markets are forecasted to grow (MarketWatch, 2022). Providing information about the human aspects of computerized systems could contribute to the ongoing public debate about them.

Theoretical background and research questions

Effects of developers' characteristics

Two streams of research have examined phenomena that are related to the aim of this study: research on algorithmic biases (Fazelpour and Danks, 2021; Kordzadeh and Ghasemaghaei, 2022), and research on software performance (Acuña, Gómez and Juristo, 2009; Gomez and Acuña, 2014). The first stream has suggested that individual characteristics, such as trust in technology and the need for cognition, could impact system adoption (Kordzadeh and Ghasemaghaei, 2022). The second stream has shown that certain personality traits (e.g. extraversion) are related to software product quality (Acuña, Gómez and Juristo, 2009; Gomez and Acuña, 2014), developer creativity (Amin *et al.*, 2020), and developer team performance (Yilmaz *et al.*, 2017).

However, research on algorithmic biases has investigated social biases such as gender and race discrimination and has not focused on other code characteristics such as speed or accuracy (Kordzadeh and Ghasemaghaei, 2022). Moreover, no study has investigated how developers' characteristics impact these social biases. Research on code performance (Acuña, Gómez and Juristo, 2009; Amin *et al.*, 2020; Gomez and Acuña, 2014) has focused on developers' big-five personality traits (extraversion, openness to experience, consciousness, agreeableness, and neuroticism) but has neglected other important characteristics such as expertise, experience, education, and error-handling. Specifically, studies about the relationship between developers' personality and code performance investigated non-professional students' performance in lab conditions (Gomez and Acuña, 2014) and hence could not yield insights on experience. However, code developers often have diverse programming experience and fields of expertise. For instance, HFT system development requires interdisciplinary knowledge of finance, mathematics, and engineering in addition to computer science expertise (Davis, Kumiega and Van Vliet, 2013). Furthermore, a priori – before conducting this study – it had been unclear which additional characteristics affected the systems. Hence, this study investigated the question:

RQ1: How do developers' characteristics affect their codes?

Effects of organizational practices

Research has shown that work practices and rules influence performance aspects in many industries (Bär, Kempf and Ruenzi, 2011; Evans, Prado and Zambrana, 2020). Similarly, in software firms, work practices affect code characteristics. For instance, code reviews[2] improve codes by detecting code failures and making them clearer and easier to modify (Mäntylä and Lassenius, 2009). In particular, formal code reviews, which follow strict rules, improve code quality significantly more than do modern code reviews, which have no guidelines (McIntosh et al., 2016). Social dominance, status, and group dynamics have been found to play major roles in software development processes, too (Bosu *et al.*, 2017; Metiu, 2006).

Although the effects of a range of firm practices and rules have been investigated, no study has examined how they influenced the extent to which developers' characteristics were reflected in their codes. However, rules could limit people's freedom to act according to their will. Therefore, this study asked whether firm rules and practices served as boundary conditions, limiting the expression of developers' characteristics in their codes:

RQ2. How do organizational practices and rules affect the reflection of developers' characteristics in their codes?

[2]Code reviews are organizational processes that aim to find defects in codes or improve them. Code reviews often comprise several stages, including a feedback meeting, defect correction, and a follow-up meeting (Ciolkowski, Laitenberger and Biffl, 2003).

Identification of developers' characteristics

The possibility that codes carry information about their developers, which enables the identification of the developers' characteristics, has not been investigated. However, in contexts other than code development, research has established that people's individual characteristics can be reliably inferred from their writings (Tskhay and Rule, 2014). For example, by reading authors' stories, people can identify some of the authors' personality traits (Küfner *et al.*, 2010). Characteristics such as impulsivity, self-esteem, and agreeableness can be inferred from social media messages of up to 140 characters (Orehek and Human, 2017; Qiu *et al.*, 2012). A few personality traits can be identified even from people's personal email addresses (Back, Schmukle and Egloff, 2008).

People's ability to identify writers' personality based on their texts has been attributed to the writers' motivation to provide cues about themselves. Research has suggested that people provide such cues both deliberately and automatically through many communication channels, including their writings (Hirschmüller *et al.*, 2013; Küfner *et al.*, 2010). Thus, people's writings encapsulate aspects of their individual characteristics.

A priori, it had been unclear whether the individual characteristics of code developers could be inferred from their codes. On the one hand, codes were often long and hence had more potential to capture information about the developers than did 140-character-long messages. On the other hand, unlike messages or stories that addressed people, computer codes consisted of variable definitions and sequences of computer commands, which were based on mathematical logic. Hence, developers' ability to express their characteristics through their codes could be limited. Thus, the third research question that this study asked was:

RQ3: Which developers' characteristics can be identified by reading their codes?

Study 1

Study 1 examined how HFT developers' characteristics and firm practices influenced HFT codes through the analysis of an interview series. As HFT uses extremely fast automated decision-making systems (Manahov, 2016) and is highly influential (Baron *et al.*, 2019; Conrad, Wahal and Xiang, 2015), it constitutes an extreme case for the investigation of the expression of developers' characteristics in their codes.

Method

A large body of research has incorporated interviews in the study of organizational and financial behaviour (D'hont, Doern and Delgado García, 2016; Sonenshein, 2014; Kellard *et al.*, 2017; Stoian, Dimitratos and Plakoyiannaki, 2018; Tilba and Wilson, 2017). Recently, it has been acknowledged that interviews could provide rich insights into HFT (Mackenzie, 2018a, 2018b). Therefore, I used interviews to study the research questions. Specifically, I used the HFT practitioner interview series of Sobolev (2020).

Interview series. Sobolev (2020) conducted a series of interviews with HFT practitioners. Each interview comprised three parts. In the first part, practitioners were asked about their daily work experience; in the second, about the way in which their personality affected their systems; and in the third, about their perceptions of the ethicality of their profession. Sobolev (2020, p. 106) used solely the third part of each interview to show that practitioners' ethics perceptions are determined by their choice of stakeholder reference groups and that their ethics perceptions influence their well-being. In this study, I drew on the second part of each interview to examine how developers' characteristics impacted their codes.

Participants. A total of 30 HFT system developers, including computer programmers, quantitative strategists, algorithm developers, traders, consultants, and managers of HFT companies, participated in the interviews. The participant group included 29 men and one woman. They worked in the UK, the USA, Sweden, Australia, or China. They were approached through LinkedIn (https://www.linkedin.com/). Participants were not paid for their participation. However, they were asked if they wanted to receive the results of the study. Additional details about the participants are given in Sobolev (2020).

Participants' professional diversity within the HFT industry fitted the aim of this study because it provided insights into a wide cross-section of system developers. Research has emphasized that practitioners from many disciplines develop HFT systems (Davis, Kumiega and Van Vliet, 2013). The number of participants fitted the aim of this study, too, as it satisfied the data saturation criterion (Guest, Bunce and Johnson, 2006; Robinson, 2014; Sobolev, 2020). Furthermore, a study concluded that it is suitable to include 30 participants in interview studies (Saunders and Townsend, 2016). Interview studies have often involved similar numbers of participants (e.g. Stoian, Dimitratos and Plakoyiannaki, 2018; Tilba and Wilson, 2017).

Procedure. A total of 28 participants were individually interviewed in face-to-face meetings, Skype video meetings (https://www.skype.com/en/), or over the phone. Two participants preferred to fill in questionnaires. Most interviews were recorded, and a few were summarized according to participants' requests. The interviews were on average 72.63 minutes long (std. dev.: 29.00 minutes; minutes: 26.30 minutes; max: 132 minutes).

The interviews were semi-structured. In semi-structured interviews, the interviewers pre-define topics or questions for discussions. However, they can fit the specific questions to each participant during the interview. Semi-structured interviews are considered natural, informative, and pleasant for the participants (Coolican, 1995).

Each interview had three parts (Sobolev, 2020, p. 106). In the second part, participants were asked whether their trading strategies or codes reflected their personalities and were given the freedom to interpret the question as they wanted. They were encouraged to provide elaborated answers. Depending on the nature of their answers, they were asked what differentiated between the strategies or codes that they and their colleagues wrote, or additional questions.

Data Analysis. I transcribed participants' answers. Then, I analysed the transcriptions using content analysis methods (Corbin and Strauss, 2008). Thus, I started by formulating concepts, describing the ideas or notions that participants expressed. I then grouped concepts into themes, and themes into dimensions, according to their contents (Gioia, Corley and Hamilton, 2013). I analysed the data using Atlas.ti, a content analysis software.

Findings

In their answers, participants referred to many individual and organizational characteristics. Hence, content analysis of the interviews yielded

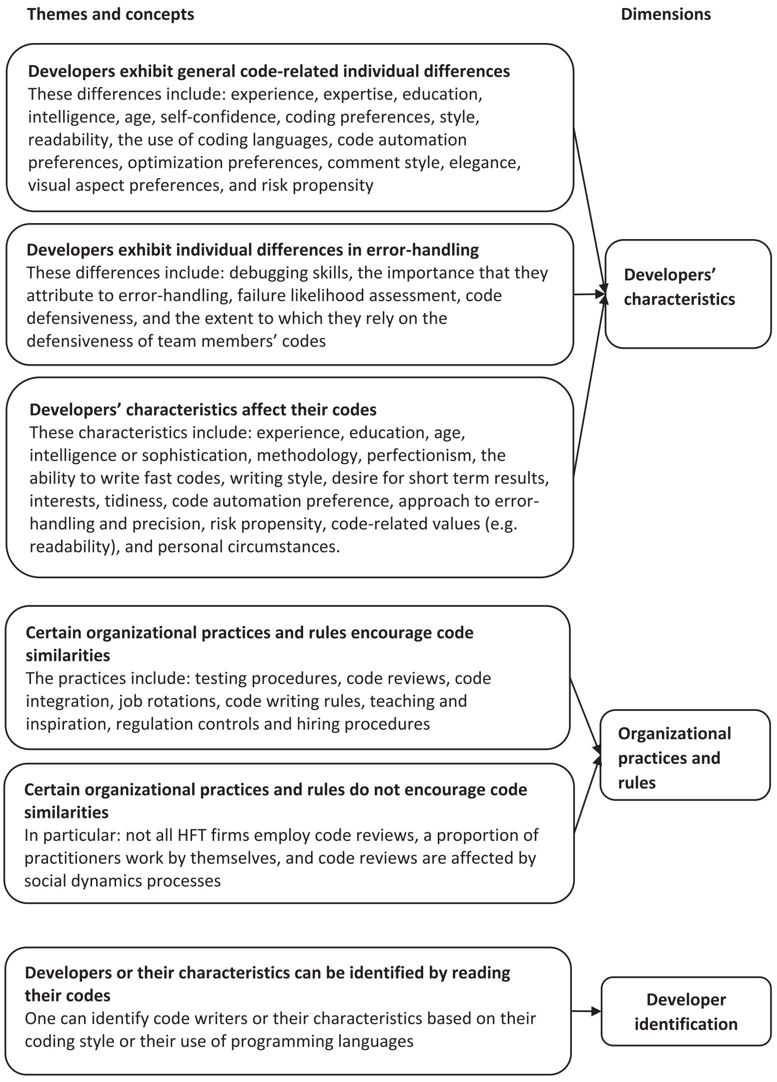
Themes and concepts
Dimensions
Developers exhibit general code-related individual differences
These differences include: experience, expertise, education, intelligence, age, self-confidence, coding preferences, style, readability, the use of coding languages, code automation preferences, optimization preferences, comment style, elegance, visual aspect preferences, and risk propensity
Developers exhibit individual differences in error-handling
These differences include: debugging skills, the importance that they attribute to error-handling, failure likelihood assessment, code defensiveness, and the extent to which they rely on the defensiveness of team members' codes
Developers' characteristics affect their codes
These characteristics include: experience, education, age, intelligence or sophistication, methodology, perfectionism, the ability to write fast codes, writing style, desire for short term results, interests, tidiness, code automation preference, approach to error-handling and precision, risk propensity, code-related values (e.g. readability), and personal circumstances.
Developers' characteristics
Certain organizational practices and rules encourage code similarities
The practices include: testing procedures, code reviews, code integration, job rotations, code writing rules, teaching and inspiration, regulation controls and hiring procedures
Certain organizational practices and rules do not encourage code similarities
In particular: not all HFT firms employ code reviews, a proportion of practitioners work by themselves, and code reviews are affected by social dynamics processes
Organizational practices and rules
Developers or their characteristics can be identified by reading their codes
One can identify code writers or their characteristics based on their coding style or their use of programming languages
Developer identification

Figure 1. Study 1: Summary of the main themes, concepts, and dimensions

three main dimensions: 'developers' characteristics', 'organizational practices and rules', and 'developer identification'. Figure 1 summarizes the main dimensions, themes, and concepts. Tables 1–3 in the Appendix present related concepts and exemplifying developer quotations. Below, I describe the concepts and themes that led to the formulation of each dimension. The data analysis and presentation format used in this study draw on qualitative studies about innovative industries, including Sobolev (2020) and Sonenshein (2014, pp. 828–832).

Developers' characteristics. The dimension '*developers' characteristics*' answered the first research question ('How do developers' characteristics affect their codes?'). It emerged through the analysis from three themes: 'developers exhibit general code-related individual differences', 'developers exhibit individual differences in error-handling', and 'developers' characteristics affect their codes' (see Figure 1). All the quotations referred to in this section are presented in Table 1 in the Appendix.

Developers exhibit general code-related individual differences. Participants expressed the belief that many developers' characteristics, which were related to their codes' performance (e.g. speed), distinguished between them. These general characteristics included the developers' experience, expertise (quotation Q1.1), education (Q1.2), intelligence (Q1.3), age or generation (Q1.4), and self-confidence (Q1.5). For instance, a senior software engineer considered experience to be an important characteristic, distinguishing between developers:

> 'It all depends on what you have experienced yourself, like, even between two seniors, you might say, well, I handle things one way, because I was once in this situation, and the best way to handle it was this […] And the other person has the opposite experience. Even [though] they are at the expert level, people who designed the language, they have arguments over what is the best way to do things […] It depends on individual experiences'.

In particular, she suggested that developers' experience influenced the speed of their codes:

> 'You can get people starting off, who will be really interested in speed, and they will straight away start looking at those techniques, and then, they can use them. But, am… [most] people starting off are writing large amounts of [slow] code'.

Participants also referred to individual differences in coding preferences, style and readability (Q1.6–Q1.8), the use of coding languages (e.g. the use of functions and structures), their code automation preferences (the extent to which developers could interfere with the codes while they were running), and the importance that they attributed to code optimization. For example, an HFT trader and programmer explained that developers differed in their readability and optimization preferences:

> 'Good people don't only care about what they are writing […] Some coders just want something that works, right, and they are just writing code at, you know, a very fast pace and they are like, you know, "let it run, oh it works, ok, fine, I am not touching that anymore." […] I would like to do that, because, it's, you know, it is easy to tick, right, "I have done it, it runs fine". But […] even if it runs right, I will try to go back into it and see whether I could optimize it and make it better, either more understandable or more reliable'.

Participants also referred to differences in their comment style, elegance (e.g. the extent to which the code was concise), and preferences regarding the visual aspects of the codes (e.g. variable and function names). In addition, they suggested that there were differences in their risk propensity. For instance, a senior software developer reported that developers differed in their code-related risk aversion:

> '[I am] reasonably risk-averse, more so than a lot of people I worked with'.

Developers exhibit individual differences in error-handling. I refer by 'code error-handling' to code error identification and correction (code debugging), the preparation of the code to unusual situations, and code testing. Error-handling is directly related to code accuracy. Many participants considered error-handling to be a central aspect of their work. However, they identified individual differences in developers' debugging skills. For instance, a lead quantitative HFT infrastructure developer stated that:

> 'Debugging is a very critical thing. It's a very specific skill set. I am not good at it. Some people are amazing debuggers'.

Participants also identified individual differences in the importance that developers attributed to error-handling (Q1.9), failure likelihood assessment (Q1.10), code defensiveness (Q1.11), and the extent to which they relied on the defensiveness of other team members' codes (Q1.12).

Developers' characteristics affect their codes. Beyond merely identifying individual differences between developers, many participants reported that their codes' performance, including the codes' speed and accuracy, reflected the developers' characteristics. Specifically, they mentioned the developers' experience (Q1.13), education (Q1.14), age or maturity (Q1.15), intelligence and sophistication (Q1.16), methodology (Q1.17), perfectionism (Q1.18), the ability to write codes that are fast (Q1.19), writing style (Q1.20 and Q1.21), the desire for short-term results (Q1.22), interests (Q1.23), tidiness (Q1.24), and code automation preferences. They perceived HFT codes to reflect also the developers' approach to error-handling and precision (Q1.25 and Q1.26). Participants also suggested that the risk level of their codes reflected developers' risk propensity, and sometimes resulted in codes that were not in line with their firms' rules (Q1.27).

It is important to note that developers' code-related values affected their codes, too. For instance, an HFT software developer stated that:

'My solution may differ […] based on my past experiences and what I value as a developer, what I think is the key attribute to a good system, compared to someone who has, maybe, grown up in a different environment with different constraints'.

In particular, several participants reported that they valued code readability. Similarly, personal circumstances, such as deadlines, as well as market conditions and opportunities, influenced the codes. For instance, to realize profit opportunities, which requires quick code adaptation, developers sometimes intentionally wrote codes that could include errors, and did not follow the regular code testing procedures (Q1.28). Thus, code-related values, personal circumstances and market conditions influenced HFT codes. Furthermore, developers' abilities affected the software's profits and losses.

Organizational practices and rules. The dimension 'organizational practices and rules' answered the second research question ('How do organizational practices and rules affect the reflection of developers' characteristics in their codes?'). It emerged from the themes: 'certain organizational practices and rules encourage code similarities' and 'certain organizational practices do not encourage code similarities' (see Figure 1). All the quotations referred to in this section are presented in Table 2 in the Appendix.

Certain organizational practices and rules encourage code similarities. A proportion of participants stated that there were similarities between codes written by different developers. Participants attributed these similarities to HFT firm practices, including testing procedures, code reviews (Q2.1), integration of codes written by independent developer teams (Q2.2), and job rotations (Q2.3). In particular, testing procedures and code reviews were often perceived as practices that limited the presence of bounded rationality elements in the codes. For example, a senior HFT software engineer highlighted the impact of testing procedures:

'Because the strategies that you develop, first of all, they are not over-night […] – it's a research process that extends over weeks or months. Sometimes, for years […] Before you [use any system] you back-test it, it's a real scientific method […] So, if there is any emotional component there, I think it is taken out by the rigor of the process'.

Moreover, some HFT developers explained that their firms had strict rules, standardizing coding styles. The use of a uniform coding style encouraged developers to adapt their coding style to that of the existing system (Q2.4). Developers also expressed the will to learn or to be inspired by other developers (Q2.5 and Q2.6). Employing methods and strategies used by others could increase their prevalence and hence homogenize HFT. Furthermore, a few participants noted that their firms regulated the risk level of the codes. In their firms, individual risk propensity could not be fully reflected in the codes (Q2.7). Finally, a few participants attributed HFT strategy and code similarities to developer homogeneity. The latter was related to HFT firm hiring procedures (Q2.8 and Q2.9).

Certain organizational practices do not encourage strategy and code similarities. In a proportion of HFT firms, developers worked by themselves and no code reviews were conducted (Q2.10). Furthermore, in firms that employed code reviews, the discussions were often affected by social dynamics (Q2.11). Therefore, code reviews often focused on superficial characteristics of the code (e.g. variable names; Q2.12) and did not affect its accuracy or speed.

Developer identification. The dimension 'developer identification' answered the third research question ('Which developers' characteristics can be identified by reading their codes?'). It was related to the theme 'developers or their characteristics can be identified by reading their codes' (see Figure 1). The quotations mentioned in this section are given in Table 3 in the Appendix.

Developers or their characteristics can be identified by reading their codes. Several participants stated that they could identify developers or their characteristics by recognizing their coding style (Q3.1) or their use of the programming languages (Q3.2). For example, a senior HFT software expert stated that he could identify code developers by reading their codes:

'You can read somebody's code and see their thumbprint on it […] You know who has written the code even if they haven't necessarily put their name on the top […] It's just like reading somebody's handwriting. If you work with somebody regularly, you know, you can see [their style]'.

Discussion

Contributing to the literature on the expression of individual differences in organizational decision-making (Adams and Jiang, 2017; Hillenbrand *et al.*, 2020) and going beyond research on algorithmic biases (Kordzadeh and Ghasemaghaei, 2022) and code characteristics (Gomez and Acuña, 2014), Study 1 suggested that HFT developers' characteristics influenced their codes in many ways. In particular, they affected the codes' speed and accuracy and could therefore distance the codes from the ideal of unbounded rationality. Although certain organizational practices, rules, and regulations limited the extent to which these individual characteristics were reflected in the codes, HFT systems carried specific information about their developers. Moreover, Study 1 suggested that developers' values and personal circumstances influenced their codes, although they could be incongruent with this ideal.

Study 2

Study 1 showed that in the HFT industry, codes reflected developers' characteristics. However, it did not examine industries other than HFT. Extending Study 1, Study 2 investigated code development in a range of companies, including additional finance, healthcare and business management firms. Furthermore, it systematically examined the differences between developers' codes and identified developers' values. In addition, it investigated the effects of personal circumstances, market conditions, and regulations. As developers' roles and familiarity with the software could influence their codes, it considered the effects of these factors, too. Study 2 consisted of an individual, online qualitative survey.

Method

Participants. Thirty-five code developers residing in the UK, the USA, or Canada participated in the study. They were recruited through Prolific (https://app.prolific.co). As three participants submitted incomplete or unusable questionnaires, the analysis drew on the answers of 32 participants. The sample satisfied the data saturation criterion (Guest, Bunce and Johnson, 2006; Robinson, 2014). It included 21 men, 10 women, and one non-binary participant. Participants' mean age was 25.91 years (std. dev.: 8.12 years, min: 19 years, max: 53 years). Five of the participants had a postgraduate degree, and 10 had an undergraduate degree. Participants worked in finance/accounting firms, medical/healthcare firms, business management/administration firms, or other firms. Accordingly, their coding outputs were diverse, including, for example, machine-learning codes for the analysis of cancer genomic data, healthcare doctor appointment websites, storefront pages for client purchases, payroll and tax calculation systems, and computer antivirus software. Participants worked as interns, software developers, senior software developers, or lead programmers at their firms. Eleven participants worked in large companies (more than 1000 employees), and eight participants worked at small firms (50 or fewer employees). Participants were paid £3.50 for their participation.

Materials. The questionnaire was designed to obtain qualitative insights about developers' code-writing processes. Therefore, participants were asked to explain their answers. The questionnaire included questions about the topics listed below.

Work description. Participants were asked to confirm that they were writing computer codes to develop software and to describe their main work responsibilities. Participants were also asked to describe the codes that they developed and to indicate the main application field of their codes.

Differences in codes, *code-related values*, *and error-handling procedures.* To enable participants to express their perceptions and avoid biasing their answers, participants were first asked whether their codes exhibited a particular style ('Do you think that your codes (design and implementation) exhibit a particular style? If yes, please describe some of the unique characteristics of your codes. If not, please explain why'). Only afterwards were they asked whether there were differences between their codes and other developers' codes, written for the same or a similar purpose, and to explain their answers. Participants were further asked whether there were differences between them and their colleagues in code-related values and in the ways in which they debugged and tested their codes.

Sources of inspiration and influences on the codes. Participants were asked to describe their sources of inspiration. They were also asked to report whether and how their codes were influenced by personal circumstances, market conditions (e.g. the COVID-19 pandemic), and regulation. In addition, participants were asked whether their firms had been involved in regulatory infractions, and whether such infractions affected their codes.

The extent to which the style of the codes reflected aspects of the developer's self. Participants were asked: 'To what extent does the style of your codes (design and implementation) reflect aspects of yourself? Please explain your answer'. Then, they were encouraged to consider their role in the firm ('Software is often developed by many individuals. Do you have a central role in the development of your firm's software as a whole? If yes, in which ways is your role central? In any case, please explain your answer') and familiarity with the details of their firms' software that they did not develop. Following these questions, participants were asked: 'Considering your role in the development of your firm's software as a whole and your knowledge of its details, are your individual characteristics reflected in the software that you work on? If yes, how? If not, does the software you work on reflect the individual characteristics of other people? How?' Participants were also asked whether their codes reflected their ethical or moral standards.

Participants' demographic details and firms' details. Participants were asked to report their gender, age, highest level of education and degree field, nationality, their firm's location (city and country), and the size of the firm, expressed as an estimate of the number of employees.

The questionnaire included additional questions. It is given in the supplementary material file.

Findings

As with Study 1, I analysed participants' answers using content analysis methods. The themes, concepts and dimensions that arose though the analysis are summarized in Figure 2. Quotations, exemplifying some of the obtained concepts are presented in Tables 4–6 in the Appendix.

Differences in developers' codes, code-related values, and error-handling procedures. All the quotations referred to in this section are presented in Table 4 in the Appendix.

Code differences. Many participants reported that their codes had a particular style, and over half of the participants indicated that there were differences between the codes that they and other developers wrote for the same or a similar purpose. Participants identified differences in their code efficiency (Q4.1), accuracy, logic, readability, user-friendliness (Q4.2), and external code characteristics, such as variable names and comment style. For example, a participant who developed data analysis software for a bioinformatics firm wrote:

> 'Yes, I think my code is significantly more readable and easy to learn from than others who may be working on accomplishing a similar task',

whereas a participant who developed buying and selling service platforms wrote:

> 'Yes, other people's code is much better, cleaner and more efficient and logical than mine'.

As with HFT practitioners, differences in code efficiency were often attributed to experience. For instance, a participant who developed data analysis software wrote:

> 'Yes, definitely. I'm not an extremely experienced programmer, so the work I do is not always extremely optimized and as efficient as possible'.

Participants who did not identify differences between their codes and others' codes often attributed this uniformity to the necessity to follow their firms' protocols, rules, or guidelines, or to similarities in developers' experiences and education. For instance, one of the participants wrote:

> 'No, we produce our code tightly to our company guidelines and everyone's style will eventually end up the same through a combination of our formatters, style guides and linters'.

Differences in code-related values. Participants had diverse code-related values, including efficiency (Q4.3), accuracy (Q4.4), computational speed, logical consistency, readability (Q4.5), ease of maintenance, flexibility, reusability, modularity, elegance, simplicity, conciseness, neatness, innovation, novelty, creativity, and uniqueness (Q4.6). For instance, one of the participants wrote:

> 'Definitely. Some people value functionality over all style and some people value style almost entirely over functionality. I very much like my code to be readable and I like stylistically nice code [...] One of my colleagues absolutely does not care for form at all'.

Differences in error-handling. About half of the participants reported that there were differences between their and other developers' debugging procedures (Q4.7) and testing procedures (Q4.8). The rest of the participants either were unsure about error-handling differences between them and other developers, or reported that there were none.

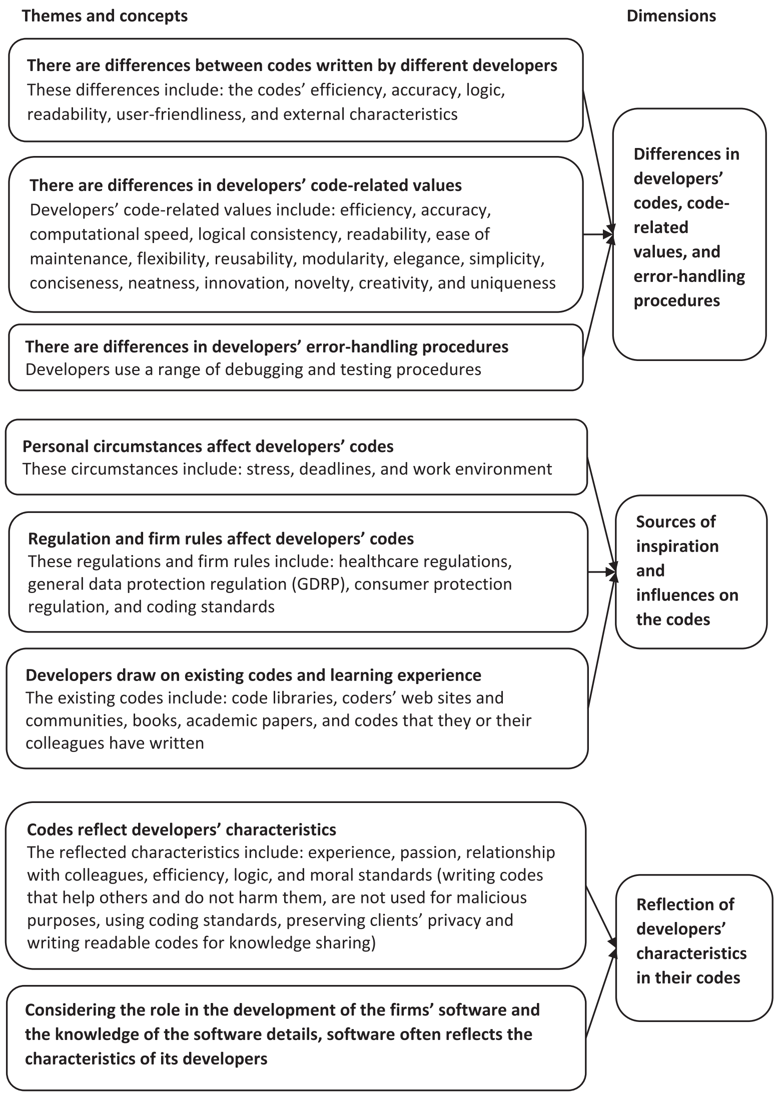
Themes and concepts
Dimensions
There are differences between codes written by different developers
These differences include: the codes' efficiency, accuracy, logic, readability, user-friendliness, and external characteristics
There are differences in developers' code-related values
Developers' code-related values include: efficiency, accuracy, computational speed, logical consistency, readability, ease of maintenance, flexibility, reusability, modularity, elegance, simplicity, conciseness, neatness, innovation, novelty, creativity, and uniqueness
There are differences in developers' error-handling procedures
Developers use a range of debugging and testing procedures
Differences in developers' codes, code-related values, and error-handling procedures
Personal circumstances affect developers' codes
These circumstances include: stress, deadlines, and work environment
Regulation and firm rules affect developers' codes
These regulations and firm rules include: healthcare regulations, general data protection regulation (GDRP), consumer protection regulation, and coding standards
Developers draw on existing codes and learning experience
The existing codes include: code libraries, coders' web sites and communities, books, academic papers, and codes that they or their colleagues have written
Sources of inspiration and influences on the codes
Codes reflect developers' characteristics
The reflected characteristics include: experience, passion, relationship with colleagues, efficiency, logic, and moral standards (writing codes that help others and do not harm them, are not used for malicious purposes, using coding standards, preserving clients' privacy and writing readable codes for knowledge sharing)
Considering the role in the development of the firms' software and the knowledge of the software details, software often reflects the characteristics of its developers
Reflection of developers' characteristics in their codes

Figure 2. Study 2: Summary of main themes, concepts and dimensions

Sources of inspiration and influences on the codes. The quotations referred to in this section are given in Table 5 in the Appendix.

In line with the ideal of unbounded rationality in codes, about half of the participants stated that personal circumstances did not affect their codes (Q5.1 and Q5.2). For example, a participant who worked on security protocols and programs for the NHS wrote:

> 'No. I try to keep my work life separate from my home'.

However, in disagreement with this ideal, other participants suggested that their personal circumstances influenced their codes. Specifically, developers highlighted that stress, deadlines (Q5.3), and work environment (Q5.4) could impact the accuracy of their codes. For instance, one of the participants wrote:

> 'My codes can be influenced by personal circumstances because sometimes it might get messy or I might make more mistakes based on if I'm in a rush to finish or I'm just stressed out'.

Most participants reported that regulations did not impact their codes. Nevertheless, several participants noted that healthcare regulation, general data protection regulation (GDRP), or consumer protection regulation impacted their codes. Participants suggested that firm rules influenced their codes, too. A single participant reported that his firm had been involved in regulatory infractions.

Most participants reported that market conditions did not affect their codes. Only few participants indicated that during the COVID-19 pandemic their workload increased due to market demand. Furthermore, during the pandemic, working from home reduced the amount of feedback that they received from others and limited the sharing of ideas.

Many developers reported that they drew their inspiration from existing codes, including code libraries, coders' websites and communities (Stack Exchange, GitHub, Reddit), books, academic papers, and codes that they had written before. Developers also mentioned their colleagues' work and the feedback they obtained from them as sources of inspiration. Learning experiences were perceived to affect the codes, too (Q5.5).

Reflection of developers' characteristics in their codes. All quotations in this section refer to Table 6 in the Appendix.

Participants exhibited diversity with respect to the reflection of their characteristics in their codes. Participants who reported that their codes did not reflect aspects of themselves attributed this to the lack of ability to express their personality (Q6.1) or the lack of a need to do so (Q6.2). However, over half of the participants reported that their codes reflected aspects of themselves, including their experience, passion (Q6.3), relationships with colleagues (Q6.4), efficiency (Q6.5), and logic (Q6.6). For instance, one of the participants expressed the idea that his codes reflected his experience as follows:

> 'I think as my code gets more automatic it reflects my growth as a programmer. Implementation-wise as it gets more efficient [it] reflects my betterment in computational thinking'.

Over a third of the participants considered their role to be central. That was especially the case for participants who were code developers or lead programmers. The rest of the participants thought that their roles were not central due to the nature of their work or because they were a part of a team (e.g. 'No, if I work in a team it's usually in an equally distributed role'). A similar proportion of participants reported that they were highly familiar with the parts of the software that they had not written. The reasons for their familiarity included code management review responsibilities (e.g. 'I manage almost all of the development my research lab does, which means I do regular code and functionality reviews. [Therefore,] I'm quite familiar with almost all of the software my lab produces'), reading code documentation and meetings (e.g. 'Yes, aware through reading documentation and hearing about work in meetings'), or training ('[Yes, because] this is a requirement when training'). Other participants reported that they had partial knowledge of the software, or that they were not familiar with it (e.g. because of non-disclosure agreements).

Considering their role in the development of the firms' software and their knowledge of the software details, a proportion of participants reported that their codes did not reflect any individual's characteristics. These participants held that the software reflected their team efforts (Q6.7) and firm rules or standards (Q6.8 and Q6.9). However, over half of the participants reported that their firms' software reflected their individual characteristics or the characteristics of other individuals. In particular, participants suggested that the software reflected their educational background (Q6.10), perfectionism (Q6.11), readability aims, error-handling and testing approach (Q6.12), efficiency, and accuracy. Furthermore, one of the participants highlighted that developers could be identified through their codes:

> 'You can usually tell who wrote what parts by the way it's written'.

Participants who suggested that the code reflected the individual characteristics of other people emphasized that it reflected the characteristics of those who had created the main parts of the software, including the team lead, the system's architect (Q6.13), colleagues, product managers (Q6.14) or their clients (Q6.15). For instance, a participant wrote:

> 'It doesn't reflect my character particularly. But, it reflects my team lead's characteristics. He is very knowledgeable and organized. Our software is mostly developed by him, so our code is so organized'.

Some of the participants asserted that their codes did not reflect their ethical or moral standards. These participants did not justify their answer (e.g. wrote only 'No'), expressed the belief that ethics had no relevance to their work (Q6.16), or transferred the moral responsibility to the user (Q6.17). For instance, one of the participants wrote:

> 'No. Code has nothing to do with one's belief'.

One of the participants indicated that there was variance in developers' moral conduct and that not all developers acted ethically (Q6.18). However, many of the participants reported that their codes reflected their ethical or moral standards. In particular, participants expressed their ethicality by writing codes that helped others and did not harm them (Q6.19), by developing codes that served their declared purposes and abstaining from using the codes for malicious purposes (Q6.20), by following coding standards such as the IEEE standard, by preserving clients' privacy, and by writing readable codes (Q6.21 and Q6.22). The latter was considered ethical because it enabled knowledge-sharing on programmers' websites. For example, one of the participants wrote:

> 'Sure. Things like developing a UI [user interface] that intentionally preys on those that are not as technologically literate would go against my morals and so I don't code that way'.

Discussion

The results of Study 2 extended many of the findings of Study 1 from the HFT industry to a wider range of industries. In particular, they highlighted that developers' characteristics (e.g. their experience) influenced their codes' properties, including the codes' efficiency. In addition, they supported this result also when developers considered their role in the organizational software development and their knowledge of software details. Personal circumstances could affect the software, too, and especially its accuracy. Furthermore, developers had diverse code-related values. Whereas some of their values (efficiency, accuracy, computational speed, and logical consistency) were in line with the ideal of unbounded rationality, other values (e.g. readability, ease of maintenance, reusability, elegance, innovation, creativity, and uniqueness) suggested that developers did not always aim to write unboundedly rational codes.

General discussion

Research on unbounded rationality has suggested that certain computerized systems 'appear as the ultimate means to reach the apex of rationality – a state in which bounded rationality is upended and rationality has become unbounded for the sake of efficiency and control'. Furthermore, it has emphasized that 'rationality is a value in and of itself' (Lindebaum and Den Hond, 2020, p. 259). Contributing to the literature on organizational decision-making (Delgado-García, De La Fuente-Sabaté and De Quevedo-Puente, 2010; Hillenbrand *et al.*, 2020), this study establishes that elements of bounded rationality are embedded in state-of-the-art computerized financial decision-making systems, such as HFT systems, as well as in codes that have applications in healthcare, business management, and other fields. These elements of inefficiency or inaccuracy arise due to developers' characteristics, which have been linked to bounded rationality, and are reflected in the codes. In addition, they arise because developers value many code characteristics other than rationality and try to optimize their codes accordingly.

Investigating how developers' characteristics influence their codes (research question 1), this study identifies many characteristics that affect system efficiency and accuracy, including code-related characteristics (e.g. the ability to write codes that run fast) and error-handling characteristics (e.g. debugging skills). Personal circumstances, such as deadlines and stress, could impact code accuracy, too. Exploring how organizational practices and rules affect the reflection of developers' characteristics in their codes (research question 2), this study suggests that testing procedures, code reviews, and code-writing rules can inhibit the expression of developers' characteristics in their codes. Furthermore, the reflection of developers' characteristics depends also on their roles. Hiring procedures can reduce developer diversity. Exploring the extent to which developers' characteristics can be identified from their codes (research question 3), this study shows that certain developers' characteristics (e.g. experience and age) can be often inferred by examining the codes' style, methods, and use of programming language.

Management applications

The results of this study could be used to improve hiring and training procedures in a wide range of technology companies that aim to develop accurate codes. Computerized system errors can harm firms, employees, and clients. For instance, HFT errors can cause mini or major flash crashes (Braun *et al.*, 2018; Kirilenko *et al.*, 2017). Similarly, certain healthcare machine-learning algorithms can lead to dangerous diagnoses (Richens, Lee and Johri, 2020). This study suggests that developers' error-handling skills and attitudes are highly diverse. Furthermore, there are differences in the extent to which developers value error-handling. Hence, code errors may be the result of inadequate hiring or training procedures. Specifically, this study highlights the need to assess job applicants' error-handling skills and perceptions and to develop appropriate training programs.

The findings of this study can also be used to improve firm practices. For instance, some HFT developers suggested that, in their firms, code reviews were influenced by personal considerations and hence were not always efficient. Participants working at other firms (e.g. management software developers) reported that deadlines and stress could impact the accuracy of their codes. Therefore, this study suggests that careful examination of firm practices could be beneficial.

Applications for the public debate on computerized systems

The media has extensively debated the use of computerized systems. For instance, in the context of HFT, an article titled 'March of the machines: The stockmarket is now run by computers, algorithms and passive managers' (The Economist, 2019) suggested that 'the rise of financial robotisation [...] raises questions about the function of markets'. It pointed out that it is essential to understand how HFT functions because 'stockmarkets are central to modern economies [...] How they operate has big implications for financial stability and corporate governance'. It concluded that it was 'significant that algorithms untethered from human decision-making are starting to call the shots'. Similarly, in the context of the healthcare industry, the media has debated whether computer diagnoses should be trusted (Ellis, 2020).

Highlighting that code developers transfer some of their characteristics to their systems, this study provides the media and the public with information about the nature of computerized systems. Furthermore, it suggests that the distinction between human and computer decision-making is blurred. As a result, it emphasizes code developers' accountability for both the positive aspects of their systems and the negative ones.

Limitations and future research

Drawing on a large body of research (Kellard *et al.*, 2017; Mackenzie, 2018b; Tilba and Wilson, 2017), this study answered the key research questions through the use of qualitative research methods. It used research recommendations (Saunders and Townsend, 2016) as well as the data saturation criterion (Guest, Bunce and Johnson, 2006; Robinson, 2014) to choose participant sample size, and aimed to formulate questions that enabled participants to express a wide range of perceptions. Nevertheless, as with any qualitative study, sample size and question formulation could influence the findings. Hence, it could be beneficial to complement this study by the use of quantitative research methods.

In both Study 1 and Study 2, participant samples included fewer women than men. This distribution is in line with the worldwide statistics, according to which over 90% of software developers are men and only about 5% of them are women (Vailshery, 2022). It is also in accord with research showing that business services favour male employment over women employment (Johan and Valenzuela, 2021), that social networks could hinder the recruitment of women (Allemand *et al.*, 2022), and that hiring committees exhibit gender biases (Mengel, 2021). However, gender homogeneity may influence code characteristics. Future research examining how gender diversity among developers influences code characteristics, and in particular its accuracy and speed, could improve computerized decision-making systems.

This study focused on developers' cognitive characteristics and on firm practices. However, research has suggested that emotional traits have an important role in organizational performance (e.g. Delgado-García, De La Fuente-Sabaté and De Quevedo-Puente, 2010). In the context of computerized decision systems, I hypothesize that developers' emotional traits could impact their ability to deal with code errors. Future research could explore this hypothesis.

Finally, regulation and its enforcement have been shown to influence practitioners' conduct (Cumming, Groh and Johan, 2018; Cumming, Hou and Wu, 2018). However, the results of this study show that only a proportion of developers believe that regulation affects their codes. That raises questions about the extent to which regulation impacts ethical code development and developers' perceptions of the regulation. Investigating these questions could have important applications in highly regulated industries, including the financial and healthcare industries.

Conclusion

This study shows that computerized systems include idiosyncrasies of their developers. Furthermore, it highlights the role of organizational practices and rules in computerized systems development. Thus, it suggests that computerized systems comprise elements of bounded rationality.

Acknowledgement

I would like to thank Professor Matrin Kilduff (The School of Management, University College London) for his helpful insights and support.

Appendix

Table 1. Study 1: Themes, concepts, and exemplifying quotations related to the dimension 'developers' characteristics'

Themes and concepts	Exemplifying quotations
Developers exhibit general code-related individual differences	***Expertise.*** Q1.1. 'On the coding side, there are very different skill levels. […] The […] proportion of people who can actually write good C++ code is actually fairly small.'
	Education. Q1.2. 'If, I would say, someone has a pure maths background, he will opt for techniques that are theoretically correct, right. Whereas when you have an engineering background, you know, not aiming for mathematical correctness, but for, you know, "it works well enough"'.
	Intelligence. Q1.3. 'The only conclusion I can think is, yes. Why would […] one person come up with a certain idea and another person not come up with an idea? Even though that they are both […] thinking about the same problem. […] There is, more or less, the same problem. So, yeah […] maybe one person is cleverer than the other person is'.
	Age or generation. Q1.4. 'Now there is a new generation of coders […] [They] come from a much stronger background. So – I learnt on the job […] When I was a kid, I didn't even know what Goldman Sachs was […] There is a new generation that are maths PhDs […] One of my colleagues has a PhD in maths from Stanford and Post-Doc in maths from Harvard […] It's a much stronger generation […] So, [HFT becomes] faster'.
	Self-confidence. Q1.5. 'Yes, my colleagues are slightly, like, more daring than me […] so, they may have new methodologies that I haven't yet from a class that I am not confident with'.
	Coding preferences. Q1.6. 'So, for example, am, I don't use open-source code, I write everything myself […] [Bank name] takes the code from open source, then they put their name on it'.
	Style. Q1.7. 'We all have our own style of writing code, and the things we are used to and things we are considering beautiful code'.
	Readability. Q1.8. 'Code, in my very strong opinion, is not just about what the computer will do, it is what's written for people to understand and improve upon it. […] I am trying to write code that people can read and can understand […] I am trying to make the code such that the relations between parts are visible […] On the other hand, I have colleagues who just write flow of many many lines of code just to achieve a goal, which they do, but then it's pretty tricky to understand how they got [to] that point […] It's a really different kind of thinking about code'.
Developers exhibit individual differences in error-handling	***The importance attributed to error-handling.*** Q1.9. 'My personality has influence on the code, from the perspective, that I have a very high commitment on correctness and stability. So, when I write code, the code must be correct. I apply a lot of techniques in order to make sure that the code is correct […] So, this is my personality. Some people, they are not like that'.

Table 1. (Continued)

Themes and concepts	Exemplifying quotations
	Failure likelihood assessment. Q1.10. 'Because people are very different, and one person's test for something may not be exactly as good as it should be […] All probabilities in code about testing are really anecdotal probabilities. They are what a person decides is […] the probability, personal probability of a failure. So […] you want to make sure that you get as many views on that as possible. Otherwise, […] they might have a […] lower probability of their interpretation of the likelihood of failure than is reality. Just because they are more risk-averse or less risk-averse person'.
	Code defensiveness. Q1.11. 'I guess mistakes are a part of the personality […] because some people would be more pessimistic than others […] So someone would think to himself "well, this kind of scenario can happen, I can code for that", other people would think: "oh no, that could never happen, that's just too unlikely, that's just too catastrophically wrong, it can't possibly happen", and they are not going to code for that. […] Defensive coding […] it's called'.
	Reliance on the defensiveness of team members' codes. Q1.12. 'Some people rely on the safety net of other people's ability to write code which will catch a failure scenario. And they lost hundred millions in five minutes, and that's a painful loss […] I guess that's another aspect of personality, isn't it, your belief that you are a part of a team, and that, maybe, that team, that you thought, is probably more defensive than you are, would bail you out'.
Developers' characteristics affect their codes	***Experience.*** Q1.13. 'You also see people's background in code as well. […] If you have got someone who has been doing a lot of coding in computer games and moved into […] high-frequency trading, then you will see tricks that they used in computer games in terms of bit flipping […] to speed things up […]. So, ammm, yeah, absolutely. So, ah, and there are other people who are used [to] very – I am going to say 'Microsoft way' of doing things […] They follow all the logical steps, and it's very ordered […] You can see it'.
	Education. Q1.14. 'Throughout my years of experience I have definitely found methods and methodology […] that I find particularly pleasing to use. So, […] I guess you could say that it reflects my personality, but it is based on what I have learnt'.
	Age or maturity. Q1.15. '[I am able to distinguish between codes written by] people who worked in HFT from people who didn't. It's – glancing, I will tell you whether this person has worked in the industry before or not, and I can get the age by 5 or 10 years, because […] there is a certain maturity that goes in code […] In the beginning […] there is a certain tendency to […] show off. And later, it's stripping down to bare essence. You know, you don't have to flourish […] I rediscovered ballet dancers recently. In the beginning it is all about raising the leg as high as possible. But then it becomes the movement of a finger, the nuance of the fingers […] It is something to do with the craft after a certain number of years'.
	Intelligence or sophistication. Q1.16. 'The code that I write […] [is] tricky […] Just one example is, you might be able to say, OK, when the market ticks up, I am going to run this calculation and decide whether I am buying. Or, someone [could] say before the market has ticked up, I am actually going to do a calculation […] to say if the market goes up, or if it stays, or it goes down, what am I going to do. So, you basically run through some scenarios in advance, so that when it happens, you have got the answers […] If you are then competing against another algorithm, which isn't doing that, well, you are going to beat it. Every time'.
	Methodology. Q1.17. 'I would say that the way I think things out […] is reflected [in my code]. I am quite methodical in my approach to things, so, I hope that [it] shows in the code as well, that it is methodical'.
	Perfectionism. Q1.18. 'I think that it reflects that I always try to do things perfectly'.
	The ability to write fast codes. Q1.19. 'Yes! Yes, there is! There is a lot of variations in development. In our company we have a lot of junior developers and senior developers. So, there will be obvious differences. [The juniors] may not know the right way or the fast way to do something […] They may not always know the performance techniques. So, there [are] slow areas in their code.'
	Writing style. Q1.20. 'Different people have different ways of, ah, writing code. And yes, you can see that, you can see that'.
	Q1.21. 'My friend told me [laughing] […] "oh you write code in so violent way" […] My friend likes elegant codes, and I like codes that go straight […] You can see clearly what I want to do in the codes […] Writing codes in C++ is art! It's just like you are painting, you are drawing a picture, it's not like [taking a] photo […] C++ is a language, it's art! […] In C++ […] some codes are really beautiful, and my code looks like a man with muscles […] If you are coding in C++ […] you can definitely tell the difference between these different code styles'.

Table 1. (Continued)

Themes and concepts	Exemplifying quotations
	The desire for short term results. Q1.22. 'I usually focus on short-term profit […] yeah, so it reflects my personality […] because I want to see the short-term results instead of waiting years. For example, in hedge funds […] you always [wait] for years or half years'.
	Interests. Q1.23. 'I really find this subject really, really fascinating. This is why I am here; this is why I crossed the ocean to be here'.
	Tidiness. Q1.24. 'I try to make my code organized. And my desk is actually pretty organized as well […] So, at least in some aspects of life I try to be organized and that includes code'.
	Approach to error-handling and precision. Q1.25. 'I would say […] it reflects my analytic side, to be very thorough with what I am doing and not making any mistakes'.
	Q1.26. 'I prefer second-order accuracy or fourth-order accuracy of the numerical method […] My friend, he said: "come on, it's the finance world! For the dynamics you just need the first order" […] Second-order accuracy […] is really good […] It's good heart'.
	Risk propensity. Q1.27. '[Some developers are like] 'cowboys' […] They […] tend to take huge risks in their algorithms even if they sometimes break the corporate policy. For example, you have a […] risk probability [of] 1%, they normally break it in their code. [Other developers] are much [more] risk-intolerant, and they are very careful'.
	Personal circumstances. Q1.28. '[Implementing] some change that a trader wants to incorporate into the system, am [costs] a certain amount of […] time and material, developer resources and so on. Now, often […] if I wasn't operating [in HFT] environment, I was just a part of the technology organization, I may say: "this is the best way to do it, and it's going to cost you three months," and I would just get on and do that work. But when you are sitting with them you see that there are opportunities which are only there today, you need to […] take those opportunities, and even if there is a risk of failure […] I always think in terms of, I want to build my software so that it will never fail, but sometimes, to build very fast software, you have to compromise and make it fragile. And it can fail. But it's ok, so long as it's working within a controlled environment, where failure is managed'.

Table 2. Study 1: Themes, concepts and exemplifying quotations related to the dimension 'organizational practices and rules'

Themes and concepts	Exemplifying quotations
Certain organizational practices and rules encourage code similarities	***Code reviews.*** Q2.1. 'There is a lot of effort put into making things consistent, lots of peer reviews, so before any change goes on to the Master […] which is the one which finds its way to production, […] there are tight controls […] People are going over it, asking questions, […] and that […] encourages people to have a consistent style'.
	Code integration. Q2.2. 'To get rid of personality in code […] you will have redundant systems written by different people […] So, for instance, I had two risk engines on my front end, one written in one language by one set of developers, and one written in another language by a different set of developers, and they were in series. So, for this very important part of the code, which is the risk engine for the execution, it had two checks by independent [codes]. […] So that's how you get rid of personality in code. But there is always personality in code'.
	Job rotations. Q2.3. 'In previous work we rotated on modules of a system a lot and as a result, different modules were similar'.
	Code writing rules. Q2.4. 'If they [code writers] are good, they should be fitting to the surrounding code, they adapt like chameleons to the code that surrounds them. They will say: "OK, on this project we use some kind of style, I will do my best to fit in"'.
	Teaching and inspiration. Q2.5. 'You learn something from somebody else, who says, "that's not a good way to do it". So, you think, oh yeah, the way you have just explained is better. So, therefore, you implement that yourself'.
	Q2.6. 'I try not to invent algorithms by myself. I always include mathematicians and quants into that, because they often have something very interesting to say and teach me a lesson or two'.

Table 2. (Continued)

Themes and concepts	Exemplifying quotations
	Regulation controls. Q2.7. ‘Crossing above your risk level in […] [bank name] is – I would say it, ah, without any doubt: impossible. The amount of different layers within the organization that prohibits you [from] going beyond that risk level is so robust […] that it’s almost boring in some ways’.
	Hiring procedures. Q2.8. ‘Typically, […] you get hired because of your similarity to the other people [who are] already in the firm. That’s a part of the culture. So […] if the hiring manager likes to risk, and does not think defensively, then the odds are that he is going to employ more and more people like that’.
	Q2.9. ‘I think, to a large degree, we are very similar personality-wise. It is a very homogeneous workplace. I mean, not [only] ethnic, even ethnic or gender-wise, but also mind-set-wise. Because […] the system you are working on – it’s a very good filter, it, kind of, homogenizes the workforce, [so that practitioners have] very specific skills and mind-sets’.
Certain organizational practices and rules do not encourage code similarities	Q2.10. ‘[In] the bank […] it was very much the case, that you would write a piece of software and it was up to you to test it and you take full responsibility. So, the code very much reflected your personality, your ability and inability to think logically’.
	Q2.11. ‘There may be egos involved [in code reviews.] […] One thing where this shows is that you think that everyone else’s code is rubbish, except for your own. […] So, when I see actually code, I say – “well, actually it’s not rubbish […] I should respect [it]”. But, in addition to that, […] I almost always want to touch someone else’s code, as in, change this or change that […] and I receive the same from the other people. Like, when they see my code, they say: “OK, it’s decent, but I will change this or change that”’.
	Q2.12. ‘It’s a pity, really, because, you know, either you [should] confess that you don’t understand [if] you are not afraid, or you [should] make sensible comments, like “oh, be aware, cause that’s going to break something, right, that functionality should not be written like that, that should be different” […] And the comments that I get are just like “oh, I don’t really like that variable name, could you make it [different]?” […] [I answer:] “OK, I will do it if it makes you more comfortable. But please pay attention to the logic”’.

Table 3. Study 1: Concepts and exemplifying quotations related to the dimension ‘developer identification’

Concepts	Exemplifying quotations
Developers or their characteristics can be identified by reading their codes	***One can identify code writers or their characteristics based on their coding style.*** Q3.1. ‘Different people have different coding styles, and […], at least [in] my team, it is very easy to tell which code is mine. The style is very clear. Maybe the whole structure […], architecture of the system, if you consider it as a building, it’s solid, it has some rules, but inside, how you decorate your own room, it’s quite different. So, I can easily tell, ok, this part of the code is written by who’.
	One can identify code writers or their characteristics based on their use of programming languages. Q3.2. ‘I can read code and tell the mood of the [code writers] […] And their state of mind […] – whether they have been interested in the code, whether they have been really safe […] For me, coding is poetry […] [I can] guess his age […] So, for example, you have got a language like Java […] There is a certain style of coding – Java promotes that coding. If you are older, like I am, I mean, I first learnt how to code in Assembly language. So, the code I write is much more like C++ than Java. So, as soon as I see a bit of code, I know [whether] this person is experienced in Java. Therefore, that person, is typically between 30 and 35. And I can tell how much experience the person has got as well, because there are certain shortcuts you make […] I can tell if someone is as old as me, because they will probably write in a similar style’.

Table 4. Study 2: Concepts and exemplifying quotations related to the dimension 'differences in developers' codes, code-related values, and error-handling procedures'

Concepts	Exemplifying quotations
There are differences between codes written by different developers	***Efficiency.*** Q4.1. 'Most of the programmers have their own quirks and approach to structuring code, and I have noticed their use of more efficient methods […] I attribute this to experience, and the length of time served within our company'.
	User-friendliness. Q4.2. 'I always see who will use this software before starting and make it easier accordingly. More user-friendly'.
There are differences in developers' code-related values	***Efficiency.*** Q4.3. 'Yes some people are more focused on making it as short / efficient as possible'.
	Accuracy. Q4.4. 'I value testing and error correcting a great deal more than most'.
	Readability. Q4.5. 'Some people don't value coherency in code as much as I do. When I code, I want new readers to be able to tell what is going on without having to spend hours looking at the code'.
	Uniqueness. Q4.6. 'I think I tend to value uniqueness and cool-factor a lot more than others [and] to make my code as interesting as possible. Innovation is extremely cool to explore when efficiency-chasing is not a priority!'
There are differences in developers' error-handling procedures	***Debugging procedures.*** Q4.7. 'Some of the senior programmers can identify the likely cause of the website or app not performing as intended simply from what does and doesn't work; I am not yet at that level. I will break up the affected section of code and identify working elements before running through line by line to debug any issues, starting with the more obvious mistakes (typos in variable or operation names, missing parentheses or semicolons etc)'.
	Testing procedures. Q4.8. 'I take a very scientific and rigorous approach to testing (unless I'm 100% logically sure that my code is correct […]). Some people don't follow the scientific method quite so closely and take the approach of just testing enough until they've convinced themselves that their code works. In any case, it just depends on how high you set the bar for your personal testing standards'.

Table 5. Study 2: Concepts and exemplifying quotations related to the dimension 'sources of inspiration and influences on the codes'

Concepts	Exemplifying quotations
Personal circumstances do not affect developers' codes	Q5.1. 'No. Regardless of what's going on in my life or how I'm feeling my code is still the generic code it takes to get the job done'.
	Q5.2. 'No, I tend to stick to more conventional coding styles which keeps me on track without much influence from my personal circumstances'.
Personal circumstances affect developers' codes	***Deadlines.*** Q5.3. 'Sometimes yeah. I guess when I am in a hurry or I have a deadline to meet, then I can't test my code thoroughly which sometimes results in poor code or error'.
	Work environment. Q5.4. 'Yes. By myself I find my code takes longer to produce, but with less mistakes. However, in a faster paced environment surrounded by my co-workers my code is produced much quicker; however mistakes are likely to slip in'.
Learning experience affects developers' codes	Q5.5. 'No, they are not influenced by personal circumstances. I'd rather say they are influenced by the progress that I have made in [my] journey on learning to code'.

Table 6. Study 2: Concepts and exemplifying quotations related to the dimension 'reflection of developers' characteristics in their codes'

Concepts	Exemplifying quotations
Codes do not reflect developers' characteristics	Q6.1. 'Not much because I don't think I'm a very good developer so it's not really an indication of aspects of myself'. Q6.2. 'Not really at all. At the end of the day, I am an engineer trying to solve a problem, not show my personality'.
Codes reflect developers' characteristics	***Passion.*** Q6.3. 'A great deal of my passion and thought-process is reflected within my software'. ***Relationship with colleagues.*** Q6.4. 'I think my coding style definitely reflects some aspects of myself. I like to make sure my peers learn and work with me, instead of desperately trying to keep up with me as I forge ahead. I'm also eager to learn, and make sure my code reflects possible improvements that can be made'. ***Efficiency.*** Q6.5. 'I like things to be efficient, simple and uncomplicated, I also prefer this in life with most things at least'. ***Logic.*** Q6.6. 'I am a very logical thinker, and prefer not to be ambiguous in both my actions and use of language, whether written or spoken. I don't rush important tasks to ensure I produce the best work that I am capable of. My code reflects my working attitude quite closely'.
Considering the role in the development of the firms' software and the knowledge of the software details, the software does not reflect developers' or other people's characteristics	Q6.7. 'No as my work is always part of a team or larger group and we work with clearly defined processes'. Q6.8. 'We have to follow [the] same rules or styles to develop any software. So it [is] basically companies characteristics that [it] reflect[s]'. Q6.9. 'No, it does not reflect individual characteristics of other people. It's an industry format'.
Considering the role in the development of the firms' software and the knowledge of the software details, the software reflects developers' or other people's characteristics	***The codes reflect the participants':*** ***Educational background.*** Q6.10. 'A lot of my methods in medicine are reflected within my software'. ***Perfectionism.*** Q6.11. 'My characteristics are reflected in the software because you can tell that I put a lot of work into it and that I made sure it [is] the best work that I could do'. ***Error-handling and testing approach.*** Q6.12. 'Yes, to an extent, as […] I would think of all possible situations that could create an error or a situation where the code would not work and fix that up before the final step, however some others just write the code because it's their job'. ***The codes reflect other individuals*** Q6.13. 'We work towards the lead architect's vision and structure, so it more reflects his characteristics'. Q6.14. 'It's a functional product that reflects the decisions of project managers, product managers and developers'. Q6.15. 'We build software for other companies, so we follow the client's requests'.
Software does not reflect the developers' moral standards	Q6.16. 'No. It does not come into play'. Q6.17. 'No, I am the writer, what they do with it is [on] them'.
Software reflects the developers' moral standards	Q6.18. 'I ensure my code is as legible and clearly documented as I can make it to enable the client to carry out alterations either by themselves or, if they desire, another development firm. I have seen several cases of poor documentation or unclear code that suggests others purposefully obscure their code from clients'. Q6.19. 'Yes, [it's] clean code that only serves to help the company and the people that access our services'. Q6.20. 'I keep security in mind at all time and do not inject anything malicious doing what the app is not supposed to do to serve my purpose / other parties' interest'. Q6.21. 'Yes, I think my code does reflect my ethical standards. I try to write for readability, to make sure knowledge is shared as much as possible'. Q6.22. 'I hope so: it's reproducible and open-source (published on GitHub)'.

References

Acuña, S. T., M. Gómez and N. Juristo (2009). 'How do personality, team processes and task characteristics relate to job satisfaction and software quality?', *Information and Software Technology*, **51**, pp. 627–639.

Adams, M. and W. Jiang (2017). 'Do chief executives' traits affect the financial performance of risk-trading firms? Evidence from the UK insurance industry', *British Journal of Management*, **28**, pp. 481–501.

Allemand, I., J. Bédard, B. Brullebaut and J. Deschênes (2022). 'Role of old boys' networks and regulatory approaches in selection processes for female directors', *British Journal of Management*, **33**, pp. 784–805.

Amin, A., S. Basri, M. Rahman, L. Z. Capretz, R. Akbar, A. R. Gilal and M. F. Shabbir (2020). 'The impact of personality traits and knowledge collection behavior on programmer creativity', *Information and Software Technology*, **128**, p. 106405.

Anderson, A. (2013). 'Trading and under-diversification', *Review of Finance*, **17**, pp. 1699–1741.

Back, M. D., S. C. Schmukle and B. Egloff (2008). 'How extraverted is honey.bunny77@hotmail.de? Inferring personality from e-mail addresses', *Journal of Research in Personality*, **42**, pp. 1116–1122.

Bär, M., A. Kempf and S. Ruenzi (2011). 'Is a team different from the sum of its parts? Evidence from mutual fund managers', *Review of Finance*, **15**, pp. 359–396.

Barber, B. M. and T. Odean (2008). 'All that glitters: the effect of attention and news on the buying behavior of individual and institutional investors', *The Review of Financial Studies*, **21**, pp. 785–818.

Baron, M., J. Brogaard, B. Hagströmer and A. Kirilenko (2019). 'Risk and return in high-frequency trading', *Journal of Financial and Quantitative Analysis*, **54**, pp. 993–1024.

Bosu, A., J. C. Carver, C. Bird, J. Orbeck and C. Chockley (2017). 'Process aspects and social dynamics of contemporary code review: insights from open source development and industrial practice at Microsoft', *IEEE Transactions on Software Engineering*, **43**, pp. 56–75.

Bozec, R., M. Dia and Y. Bozec (2010). 'Governance–performance relationship: a re-examination using technical efficiency measures', *British Journal of Management*, **21**, pp. 684–700.

Braun, T., J. A. Fiegen, D. C. Wagner, S. M. Krause and T. Guhr (2018). 'Impact and recovery process of mini flash crashes: an empirical study', *Plos One*, **13**, pp. 1–11.

Chaboud, A. P., B. Chiquoine, E. Hjalmarsson and C. Vega (2014). 'Rise of the machines: algorithmic trading in the foreign exchange market', *The Journal of Finance*, **69**, pp. 2045–2084.

Ciolkowski, M., O. Laitenberger and S. Biffl (2003). 'Software reviews: the state of the practice', *IEEE Software*, **20**, pp. 46–51.

Conrad, J., S. Wahal and J. Xiang (2015). 'High-frequency quoting, trading, and the efficiency of prices', *Journal of Financial Economics*, **116**, pp. 271–291.

Coolican, H. (1995). *Introduction to Research Methods and Statistics in Psychology*. London: Hodder & Stoughton.

Corbin, J. and A. Strauss (2008). *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. London: Sage.

Croce, A., E. Ughetto and M. Cowling (2020). 'Investment motivations and UK business angels' appetite for risk taking: the moderating role of experience', *British Journal of Management*, **31**, pp. 728–751.

Cumming, D., A. P. Groh and S. Johan (2018). 'Same rules, different enforcement: market abuse in Europe', *Journal of International Financial Markets, Institutions and Money*, **54**, pp. 130–151.

Cumming, D., W. Hou and E. Wu (2018). 'Exchange trading rules, governance, and trading location of cross-listed stocks', *The European Journal of Finance*, **24**, pp. 1453–1484.

Davis, M., A. Kumiega and B. Van Vliet (2013). 'Ethics, finance, and automation: a preliminary survey of problems in high frequency trading', *Science and Engineering Ethics*, **19**, pp. 851–874.

Delgado-García, J., J. De La Fuente-Sabaté and E. De Quevedo-Puente (2010). 'Too negative to take risks? The effect of the CEO's emotional traits on firm risk', *British Journal of Management*, **21**, pp. 313–326.

D'hont, L., R. Doern and J. Delgado García (2016). 'The role of friendship in the formation and development of entrepreneurial teams and ventures', *Journal of Small Business and Enterprise Development*, **23**, pp. 528–561.

Eisenbeiß, S. and S. Boerner (2013). 'A double-edged sword: transformational leadership and individual creativity', *British Journal of Management*, **24**, pp. 54–68.

El-Bouri, W. K., Y. X. Gue and G. Y. H. Lip (2021). '"Rise of the machines": the next frontier in individualized medicine', *Cardiovascular Research*, **117**, pp. e129–e131.

Ellis, R. (2020). 'As artificial intelligence proves as effective as doctors at reading scans, we ask… would you trust a computer to diagnose your illness?', *Daily Mail*. Available at https://www.dailymail.co.uk/health/article-8067601/Would-trust-computer-diagnose-illness.html.

Evans, R. B., M. P. Prado and R. Zambrana (2020). 'Competition and cooperation in mutual fund families', *Journal of Financial Economics*, **136**, pp. 168–188.

Fazelpour, S. and D. Danks (2021). 'Algorithmic bias: senses, sources, solutions', *Philosophy Compass*, **16**, pp. 1–16. Available at https://compass.onlinelibrary.wiley.com/doi/full/10.1111/phc3.12760.

Galouchko, K., A. Torsoli and J. Ekblom (2022). 'Citi's painful flash crash highlights risks from algo trades', *Bloomberg*. Available at https://www.bloomberg.com/news/articles/2022-05-03/citi-s-painful-flash-crash-highlights-market-risks-from-algos.

Gioia, D. A., K. G. Corley and A. L. Hamilton (2013). 'Seeking qualitative rigor in inductive research', *Organizational Research Methods*, **16**, pp. 15–31.

Gomez, M. N. and S. T. Acuña (2014). 'A replicated quasi-experimental study on the influence of personality and team climate in software development', *Empirical Software Engineering: An International Journal*, **19**, pp. 343–377.

Guest, G., A. Bunce and L. Johnson (2006). 'How many interviews are enough? An experiment with data saturation and variability', *Field Methods*, **18**, pp. 59–82.

Hajli, N., U. Saeef, M. Tajvidi and F. Shirazi (2022). 'Social bots and the spread of disinformation in social media: the challenges of artificial intelligence', *British Journal of Management*, **33**, pp. 1238–1253.

Hillenbrand, C., A. Saraeva, K. Money and C. Brooks (2020). 'To invest or not to invest? The roles of product information, attitudes towards finance and life variables in retail investor propensity to engage with financial products', *British Journal of Management*, **31**, pp. 688–708.

Hirschmüller, S., B. Egloff, S. Nestler and M. D. Back (2013). 'The dual lens model: a comprehensive framework for understanding self–other agreement of personality judgments at zero acquaintance', *Journal of Personality and Social Psychology*, **104**, pp. 335–353.

Hoffmann, P. (2014). 'A dynamic limit order market with fast and slow traders', *Journal of Financial Economics*, **113**, pp. 156–169.

Isidore, C. (2019). '*Machines are driving Wall Street's wild ride, not humans*', CNN Business (February 6). Available at https://money.cnn.com/2018/02/06/investing/wall-street-computers-program-trading/index.html.

Jarnecic, E. and M. Snape (2014). 'The provision of liquidity by high-frequency participants', *The Financial Review*, **49**, pp. 371–394.

Johan, S. and P. Valenzuela (2021). 'Business advisory services and female employment in an extreme institutional context', *British Journal of Management*, **32**, pp. 1082–1096.

Kawa, L. (2018). '*Goldman warns the rise of the machines leaves markets exposed*', Bloomberg (May 23). Available at https://www.bloombergquint.com/markets/goldman-warns-the-rise-of-the-machines-leaves-markets-exposed#gs.3rue3o.

Kellard, N., Y. Millo, J. Simon and O. Engel (2017). 'Close communications: hedge funds, brokers and the emergence of herding', *British Journal of Management*, **28**, pp. 84–101.

Kirilenko, A., A. S. Kyle, M. Samadi and T. Tuzun (2017). 'The flash crash: high-frequency trading in an electronic market', *The Journal of Finance*, **72**, pp. 967–998.

Kiss, H. J., I. Rodriguez-Lara and A. Rosa-Garcia (2020). 'Does response time predict withdrawal decisions? Lessons from a bank-run experiment', *Review of Behavioral Finance*, **12**, pp. 200–222.

Kordzadeh, N. and M. Ghasemaghaei (2022). 'Algorithmic bias: review, synthesis, and future research directions', *European Journal of Information Systems*, **31**(3), pp. 388–409. Available at https://www.tandfonline.com/doi/full/10.1080/0960085X.2021.1927212.

Koumakhov, R. and A. Doud (2021). 'Decisions and structures: a dialogue between Herbert Simon and critical realists', *British Journal of Management*, **32**, pp. 1404–1420.

Küfner, A. C. P., M. D. Back, S. Nestler and B. Egloff (2010). 'Tell me a story and I will tell you who you are! Lens model analyses of personality and creative writing', *Journal of Research in Personality*, **44**, pp. 427–435.

Lindebaum, D. and F. den Hond (2020). 'Insights from "the machine stops" to better understand rational assumptions in algorithmic decision making and its implications for organizations', *The Academy of Management Review*, **45**, pp. 247–263.

MacKenzie, D. (2018a). 'Material signals: a historical sociology of high-frequency trading', *The American Journal of Sociology*, **123**, pp. 1635–1683.

MacKenzie, D. (2018b). ''Making', 'taking' and the material political economy of algorithmic trading', *Economy and Society*, **47**, pp. 501–523.

Manahov, V. (2016). 'Front-running scalping strategies and market manipulation: why does high-frequency trading need stricter regulation?', *The Financial Review*, **51**, pp. 363–402.

Mäntylä, M. V. and C. L. Lassenius (2009). 'What types of defects are really discovered in code reviews?', *IEEE Transactions on Software Engineering*, **35**, pp. 430–448.

MarketWatch. (2022). 'High-frequency trading market growth 2022 to 2027'. Available at https://www.marketwatch.com/press-release/high-frequency-trading-market-growth-2022-to-2027-share-global-industry-size-trends-emerging-factors-demands-key-players-emerging-technologies-and-potential-of-industry-2022-03-28.

McIntosh, S., Y. Kamei, B. Adams and A. E. Hassan (2016). 'An empirical study of the impact of modern code review practices on software quality', *Empirical Software Engineering*, **21**, pp. 2146–2189.

Metiu, A. (2006). 'Owning the code: status closure in distributed groups', *Organization Science*, **17**, pp. 418–435.

Mengel, F. (2021). 'Gender bias in opinion aggregation', *International Economic Review*, **62**(3), pp. 1055–1080.

Moore, E. (2013). 'Men or machines: who runs the markets?', *Financial Tmes*, (March 15). Available at http://ig-legacy.ft.com/content/15f4631c-486d-11e2-a1c0-00144feab49a#axzz5jl6NEYAM

Nguyen, T. M. and A. Malik (2022). 'A two-wave cross-lagged study on AI service quality: the moderating effects of the job level and job role', *British Journal of Management*, **33**, pp. 1221–1237.

Orehek, E. and L. J. Human (2017). 'Self-expression on social media: do tweets present accurate and positive portraits of impulsivity, self-esteem, and attachment style?', *Personality and Social Psychology Bulletin*, **43**, pp. 60–70.

Patterson, S. and A. Osipovich (2020). 'High-frequency traders feast on volatile market: profits climb sharply with help from sophisticated computer algorithms and strategies that take advantage of rips and dips; 'a quarter for the record books"', *Wall Street Journal*, Available at https://www.wsj.com/articles/high-frequency-traders-feast-on-volatile-market-11585310401.

Qiu, L., H. Lin, J. Ramsay and F. Yang (2012). 'You are what you tweet: personality expression and perception on Twitter', *Journal of Research in Personality*, **46**, pp. 710–718.

Richens, J. G., C. M. Lee and S. Johri (2020). 'Improving the accuracy of medical diagnosis with causal machine learning', *Nature Communications*, **11**, pp. 1–9.

Robinson, O. C. (2014). 'Sampling in interview-based qualitative research: a theoretical and practical guide', *Qualitative Research in Psychology*, **11**, pp. 25–41.

Saunders, M. N. K. and K. Townsend (2016). 'Reporting and justifying the number of interview participants in organization and workplace research', *British Journal of Management*, **27**, pp. 836–852.

Simon, H. A. (1947/1997). *Administrative Behavior. A Study of Decision-Making Processes in Administrative Organizations*. New York, NY: Free Press.

Sobolev, D. (2020). 'Insider information: the ethicality of the high frequency trading industry', *British Journal of Management*, **31**, pp. 101–122.

Sonenshein, S. (2014). 'How organizations foster the creative use of resources', *Academy of Management Journal*, **57**, pp. 814–848.

Stoian, M., P. Dimitratos and E. Plakoyiannaki (2018). 'SME internationalization beyond exporting: a knowledge-based perspective across managers and advisers', *Journal of World Business*, **53**, pp. 768–779.

Tilba, A. and J. Wilson (2017). 'Vocabularies of motive and temporal perspectives: examples of pension fund engagement and disengagement', *British Journal of Management*, **28**, pp. 502–518.

Tskhay, K. O. and N. O. Rule (2014). 'Perceptions of personality in text-based media and OSN: a meta-analysis', *Journal of Research in Personality*, **49**, pp. 25–30.

The Economist. (2019). 'March of the machines: the stockmarket is now run by computers, algorithms and passive managers'. Available at https://www.economist.com/briefing/2019/10/05/the-stockmarket-is-now-run-by-computers-algorithms-and-passive-managers.

Vailshery, S. J. (2022). 'Software developer gender distribution worldwide as of 2021', *Statista*, Available at https://www.statista.com/statistics/1126823/worldwide-developer-gender/.

Wang, Y., L. Kung, S. Gupta and S. Ozdemir (2019). 'Leveraging big data analytics to improve quality of care in healthcare organizations: a configurational perspective', *British Journal of Management*, **30**, pp. 362–388.

Yilmaz, M., R. V. O'Connor, R. Colomo-Palacios and P. Clarke (2017). 'An examination of personality traits and how they impact on software development teams', *Information and Software Technology*, **86**, pp. 101–122.

Zhou, W. and N. Olivari (2013). 'EBS take new step to rein in high-frequency traders', *Reuters Newswire*, August 23. Available at https://www.reuters.com/article/us-markets-forex-hft-idUSBRE97M0YJ20130823.

Daphne Sobolev has a PhD in Psychology and a PhD in Applied Mathematics. She works as a Lecturer (Education) and an Honorary Research Associate at the School of Management, University College London. Daphne specializes in decision-making, behavioural finance, and ethics. Her work has been published in leading journals, including the *British Journal of Management* and *Risk Analysis: An International Journal*. At the School, Daphne teaches behavioural finance and leads the Master in Finance dissertation module.

Supporting Information

Additional supporting information can be found online in the Supporting Information section at the end of the article.

Supporting Information