The Journal of Systems & Software 192 (2022) 111427

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Impact of individualism and collectivism cultural profiles on the behaviour of software developers: A study of stack overflow☆

Elijah Zolduoarrati, Sherlock A. Licorish *, Nigel Stanger

Department of Information Science, University of Otago, Dunedin, New Zealand

ARTICLE INFO

Article history:
Received 13 December 2021
Received in revised form 22 June 2022
Accepted 27 June 2022
Available online 28 June 2022

Keywords:
Cultural dimension
Software development communities
Individualism
Collectivism

ABSTRACT

Literature is scarce on culture and its impact on the behavioural patterns within software development communities. However, globalisation in software development has intensified the need for software development teams to navigate culture issues to ensure the successful implementation of projects. Therefore, the current study examines whether the effects of culture on software developers conform to Hofstede's individualism cultural dimension. Individualism is studied because of its established negative impacts on teamwork, which is central to software development. Data comprised artefacts from Stack Overflow, a popular online programming community. Developers were from the United States (US), China, and Russia, three countries that differ in terms of their individualistic or collectivistic cultures. Data mining techniques, as well as statistical, linguistic, and content analysis were used to compare the orientation, attitudes, interaction, and knowledge sharing patterns of the three groups of developers. Differences revealed among the three groups were consistent with their cultural backgrounds. US developers, who are from a more individualistic culture, had higher average reputations, used the pronoun "I" more frequently, and were more task-focused. Conversely, Chinese developers, who are from a more collectivistic culture, used the pronouns "we" and "you" more frequently, and were more likely to engage in information exchange. Russian developers had been using Stack Overflow the longest and were the most reflective. The cultural patterns identified in this study have implications for enhancing in-group interactions and team behaviour management during software development, especially when global teams assemble.

1. Introduction

An understanding of culture is vital to studying information technology (IT) and software engineering because culture affects the manner in which individuals and teams operate, the successful implementation of projects, and the managerial processes governing these projects (Glisson and James, 2002). Resolving cultural issues in software engineering is essential to ensure adequate team performance when team members originate from different cultures (Leidner and Kayworth, 2006), and the importance of resolving such issues has increased with rising globalisation in software development (Marinho et al., 2018).

This study explores cultural aspects of communication among software developers utilising data from the Stack Overflow online platform. Specifically, the data were used to create measures of the orientation, attitudes, interaction, and knowledge sharing patterns of developers. These measures were then studied for contributors from three different countries that are said to diverge on the culture scales (Hofstede, 2011). *Orientation* in this work is conceptualised as a developer's quantitative profile of contributions, consisting of 11 attributes such as the length of their linguistic "About Me" profile, comments, history, questions, answers, and so on, inspired by multiple studies (Bachschi et al., 2020; Tian et al., 2019). Thus, orientation here is assessed as the way that software practitioners contribute their social and intellectual capacity to the software development community. *Attitude* is defined as the personality traits or psycholinguistic profiles revealed during the study of developers' linguistics usage on Stack Overflow. Other studies have used linguistic analysis to determine personality traits of developers in order to reveal potential patterns of attitudes (e.g., Bazelli et al. (2013) and Pennebaker and King (1999)). Overall, we sought to identify attitude patterns associated with the cultures of the software developers — patterns displayed in terms of how developers behave in the software development community that might inform future policy and management approaches for teams comprising members from diverse cultures (Oishi et al., 2021). This is an approach that other scholars have taken in similar studies

☆ Editor: Kelly Blincoe.

* Corresponding author.
E-mail addresses: elijah.zolduoarrati@otago.ac.nz (E. Zolduoarrati), sherlock.licorish@otago.ac.nz (S.A. Licorish), nigel.stanger@otago.ac.nz (N. Stanger).

https://doi.org/10.1016/j.jss.2022.111427
0164-1212/

(e.g., Oliveira et al. (2016)). The dimensions studied included 13 characteristics, such as individualistic processes, collectivistic processes, certainty processes, and leisure processes (Licorish and MacDonell, 2013b). *Interaction and knowledge sharing* pertain to the willingness of forum participants to exchange knowledge and expertise with each other as well as the nature of information exchanged. Interaction and knowledge sharing thus represent the activity of capturing, storing, sharing, and using knowledge among individuals in a community. Following established guidelines, the content for interaction and knowledge sharing was grouped into 13 behavioural attributes, including commenting, discussing, scaffolding, and so on Licorish and MacDonell (2014b).

To explore potential cultural differences among software developers, the measures were analysed separately for contributors from three countries – China, Russia, and the US – that differ in terms of Hofstede's dimension of individualism. Hofstede (1984) claimed that the extent to which individuals in a culture value individualism or collectivism is a measure of how much members depend on one another. The three nations selected have been scored out of 100 on the individualism–collectivism dimension using the Hofstede Insights country comparison tool (Hofstede et al., 2005) and the Culture Compass survey tool.[1] The US scored 91 (highly individualistic), Russia scored 39 (moderately individualistic), and China scored 20 (low). Accordingly, the three countries may be considered to span the individualistic–collectivistic continuum and studying them provides useful insights into this issue. This is particularly noteworthy when considering that all meaningful software systems are developed by teams, and teams are increasingly global in nature (Ali, 2020; Deshpande et al., 2010). Cultural awareness would aid in planning for the nuances of developers, as well as with assembling and supporting high-performing, inclusive software development teams. To address this opportunity, our exploration was guided by three research questions:

RQ1: What is the orientation of software development practitioners, on Stack Overflow, from collectivistic cultures compared to that of their counterparts from individualistic cultures?

RQ2: Do the attitudes of software development practitioners, on Stack Overflow, from collectivistic cultures differ from those who come from individualistic cultures? If so, what is the magnitude of these differences?

RQ3: How does the process of interaction and knowledge sharing compare between software development practitioners, on Stack Overflow, from collectivistic cultures versus those from individualistic cultures?

By answering these questions, this study makes several significant contributions. It adds to the literature focused on cultural and behavioural-related factors in software development. It also expands on the literature that explores how practitioners interact through the use of software artefacts. Moreover, this study provides practical contributions that can be used to better understand software developers' orientations, attitudes, and interaction and knowledge sharing patterns in relation to their particular culture. Additionally, methodological recommendations are provided for those who wish to extract and examine software artefacts. In-depth understandings of multi-method research design are also provided. Finally, this study provides insights to guide future research on software development practices.

In the next section, we examine cultural theories, cultural considerations in software development organisations, and the treatment of culture in this study. Thereafter, we discuss our methodology, which covers our data collection, measures, and data analysis. We next outline the results, before discussing these outcomes and their implications. Finally, we consider the limitations of the study and provide concluding remarks highlighting directions for future research in this area.

2. Background and related work

In this section, we discuss the theoretical underpinnings for the current study. Specifically, we explain the cultural dimensions of Hofstede (1984) and explain how these dimensions relate to the measures examined in this research. We also review research on the culture of software development organisations and related work.

2.1. Hofstede's theory of cultural dimensions

Although culture is invisible, it is nevertheless widely felt and experienced (Wickberg, 2007), and understanding culture requires diverse knowledge of human behaviour (Graziotin et al., 2021). Over the years, many cultural theories from various disciplines have emerged to define constructs for mapping and comparing different cultures (Schwartz, 2006). The cultural theory proposed by Hofstede is one of the most widely known theories of culture today, especially in the field of business management (Hofstede, 1984). Other notable approaches include the cultural theory of materialism and post-materialism from Inglehart (1981) and the individual value differences theory from Schwartz (2006). However, this literature review concentrates on Hofstede's theory of cultural dimensions because of its prominence in the management and information technology fields, as well as its utility in detecting cultural patterns (Harada, 2017; Hofstede, 2011).

Hofstede's theory of cultural dimensions provides a way of determining the relationships between culture and management. Management is defined as the set of interrelated processes for creating corporate policy and organising, planning, controlling, and directing organisational resources to achieve set objectives (Patrick and Kumar, 2012; Rahman et al., 2020). While management is a crucial aspect of human systems, culture is an essential factor for determining how individuals or specific groups of people respond to management. According to Hofstede (1984), for organisations to be successful, they must be well managed. He posited that effective management entails a deep understanding of the cultural identities of the people who are managed. Thus, the cultural aspect of management is an excellent determinant of the level of success of a given management style. Hofstede concluded that no one method of management is universally appropriate, and the choice of management style depends on many factors, with culture being a significant driving force.

In developing his theory of cultural dimensions, Hofstede analysed cultural dimensions among IBM employees across more than fifty countries (Hofstede, 2011). Through his analysis, Hofstede identified the following four value dimensions along which cultural values could be assessed: (1) individualism versus collectivism, (2) small power distance versus large power distance, (3) weak uncertainty avoidance versus strong uncertainty avoidance, and (4) masculinity versus femininity. These four dimensions are universal and can be used to describe the impact of culture on the values of specific groups, including countries (Hofstede et al., 2005). The cultural dimensions may also explain why groups behave as they do (Harada, 2017).

In terms of how Hofstede's four dimensions impact the values of a group (Hofstede, 2011), first, the extent to which individuals in a culture value individualism or collectivism is a measure

[1] https://www.hofstede-insights.com/country-comparison/china,russia,the-usa/.

of how much members depend on one another. Second, power distance depends on the degree to which a culture's members agree that power is unequally distributed. Members of societies with a large power distance willingly accept segregation of people into different social classes, while those of countries with a small power distance seek to ensure power is equally distributed among all people. Third, uncertainty avoidance refers to the degree of tolerance a society has for ambiguity. Countries with strong uncertainty avoidance maintain rigid principles, while those with weak uncertainty avoidance have more relaxed guidelines for their citizens. Finally, a masculine country is one that places great emphasis on material wealth, assertiveness, and heroic deeds, whereas a feminine country places emphasis on relationships, modesty, and quality of life (Hofstede et al., 2005). A country's placement on these four dimensions determines what type of management system will be effective in that country. Management functions of organisations are culturally constrained, and a management style effective in one country may fail to produce tangible results in another. Thus, management techniques need to align with the culture in which an organisation operates (Hofstede et al., 2005).

Hofstede (2011) later refined the original cultural dimensions theory to add two further dimensions that apply to both nations and organisations: long/short-term orientation and indulgence/restraint. Long- and short-term orientation is the degree to which a culture prepares its members to embrace the delayed gratification of their needs. For instance, in business organisations, Asian business managers are generally future-oriented and value long-term investments, while North American business managers tend to prefer short-term profitability (Hofstede, 2011). In practice, companies in North American and Asian countries evaluate their employees' performance based on long- and short-term orientations (Hofstede et al., 2005). Indulgence/restraint refers to the extent to which a culture allows gratification and the embracing of natural basic human desires for the enjoyment of life. Organisations showing restraint limit gratification and place restrictions on individuals' joyful behaviour (Fischer et al., 2010). For instance, Chinese and Indian business organisations tend to have low indulgence scores, indicating active cultural restraint where rules take precedence over freedom of speech (Hofstede et al., 2005).

Hofstede emphasised that although these dimensions can be used to explain a national culture in aggregate, individual people tend to act based on their past life experiences. This is especially true when individuals are not in their native country and are interacting with other cultures (Hofstede, 2011). Although Hofstede's research on organisational culture has been heavily criticised for its focus on a single information technology company, he was nevertheless able to show how cultural groupings influence the behaviours of people across societies and organisations, and the groupings that are persistent across disciplines (Achinivu, 2017). Therefore, Hofstede's framework of cultural dimensions, and especially the individualism dimension, was utilised in this study to analyse the orientation, attitudes, interaction, and knowledge sharing patterns of software development groups.

2.2. Culture of software development organisations

Like countries, many software proprietors have their own culture that guides their behaviours and decision-making processes (Smite et al., 2019). For instance, Spotify, a Swedish software engineering company, believes that a significant responsibility of leaders is to offer leadership and guidance to their community members and then allow the members to determine how to complete the assigned tasks (Smite et al., 2019). Google, on the other hand, believes strongly in innovation, and its management community has put many mechanisms in place to ensure employees maximise their skills in innovation (Steiber and Alänge, 2013; Wohlrab et al., 2020).

Culture plays an essential role for individuals and teams in the global software development process. A qualitative study carried out by Vasilescu et al. (2015) explored the collaboration processes of culturally distributed developers in Open Source Software (OSS) communities and found that diversity among communities and group members was crucial for developers' success when operating in virtual cooperative settings. Also, as developers in OSS communities became aware of the importance of cultural diversity, they welcomed the variety of perspectives provided on such platforms and benefited from this diversity. Deshpande et al. (2010) examined cultural diversity in global software development communities and reported the need to utilise concrete strategies to manage cultural differences in a geographically distributed software development environment. Moreover, Herbsleb and Moitra (2001) found several significant challenges inhibiting the success of global software development communities, including cultural challenges, poor communication, knowledge management issues, project and process management problems, and technical issues. Similarly, Holmstrom et al. (2006) examined challenges associated with global software development and identified cultural issues as a top difficulty facing software development professionals, after temporal and geographical issues. Language barriers, for example, made interpreting and creating shared meaning difficult.

One strategy for addressing cultural challenges within organisations is to ensure there is cultural diversity within programming groups and communities, which research has suggested will benefit the process of software development. For instance, Casado-Lumbreras et al. (2011) investigated the impact of culture on mentoring relationships within the software engineering industry. Using interviews, they obtained data from 45 mentored developers where from regions such as Latin America, the Maghreb, and Eastern Europe (including Russia). There were two significant findings from the study. First, cultural differences affected both formal and informal mentoring. Second, technical competencies did not improve after the implementation of a mentoring relationship. Similar findings from Patel (2017) suggest that cultural differences impact software development and should be considered when outsourcing offshore. Cultural factors to consider include a cultural understanding of outsourcing, shared values and norms, the cultural intelligence level of the client and the vendor, the management of cultural differences, and language barriers. Neglecting these cultural factors can unfavourably impact a project. For instance, developers from some cultures might consider reporting to a female manager to be inappropriate. In this case, an organisational hierarchy needs to be implemented strictly so that developers from such cultures accept requests and instructions coming from a senior figure regardless of their gender (Casey, 2010). In general, managing developers' attitudes can be challenging for managers, who must navigate cultural and technical factors influencing the development process. Therefore, an awareness of cultural differences and how these interplay with and impact the way individuals operate could be significant to the success of projects and communities. This is particularly necessary in light of the prevalence of increasingly globalised teams operating in the software development industry (Jaakkola and Heimburger, 2009). We further justify the need for such a research agenda in the next section.

2.3. Delineating the study of culture

Numerous studies have demonstrated how an understanding of culture and behaviour enables researchers to comprehend

and analyse the underlying behavioural mechanisms in cross-cultural contexts (Kalliamvakou et al., 2014). However, literature is scarce on the study of culture in the context of software development, particularly in terms of detecting behavioural patterns in specific development communities. The current study seeks to fill this gap by applying objective measures obtained from software development artefacts to determine the effects of culture on the behaviours of software practitioners and to investigate whether these effects conform to Hofstede's individualism cultural dimension.

This study focuses specifically on the individualism dimension because it can be used to describe an individual's interdependence (self-concept) and profoundly influences the degree of collaboration within a society (Hofstede et al., 2005). In an individualistic culture, people tend to take care of themselves and their families, while in a collectivistic culture, individuals look after a larger group and work towards long-term objectives (Hofstede, 1984). Hofstede described societies with a dominant individualistic mentality as "self-oriented" cultures since they stress individual needs over the needs of a group, and members are perceived as independent with autonomous behaviour. Conversely, societies with a collectivistic mentality rely on a hierarchical group structure to make decisions and consider group outcomes a priority. In collectivistic societies, each member shares responsibility to ensure the harmony of their community as a lack of member participation may indicate group failure (Watkins and Liu, 1996).

Hofstede (1984) indicated that a country's rank on the individualism cultural dimension reflects how it perpetuates traditional values. In a collectivistic society, in which members seek to be consultative and participative, the principles of devotion, dedication, and permanence are highly valued. People from collectivistic societies tend to be non-confrontational when making enquiries (Vallaster, 2005), making them less orientated towards direct contact. In these societies, direct contact might be viewed as a threat to group cohesion that could alter the cordial atmosphere. Conversely, members of individualistic cultures seek more immediate resolution to issues without consulting others (Ali et al., 1997).

Erez and Earley (1993) proposed that people from collectivistic cultures lean more towards indirect communication. Therefore, we would expect software developers from collectivistic cultures to be more inclined to facilitate decision-making processes. In contrast, software developers from individualistic cultures would be more likely to stress the importance of the tasks at hand over group dynamics. To members of a collectivistic culture, the approach taken by developers from an individualistic culture may seem confrontational and overly direct. Based on the existing literature (Gudykunst and Ting-Toomey, 1988; Teeni, 2001), this study postulates that software development community members from collectivistic cultures may show a higher degree of group orientation than community members from individualistic cultures. The latter are likely to be more self-oriented and to stress the needs of the individual over those of the group. RQ1: *What is the orientation of software development practitioners, on Stack Overflow, from collectivistic cultures compared to that of their counterparts from individualistic cultures?* allows us to explore this issue.

Prior research has explored the cross section between behavioural and technical factors in software development. Weinberg (1972) and Shneiderman (1980) investigated the relationship between developers' behavioural facets and performance by exploring programming as a social activity and assessing personality factors such as introversion and extraversion that impacted programming outcomes. Their findings have contributed to our understanding of community empowerment (Bird et al., 2011) and provided awareness of the behavioural factors influencing software development. In another study of the link between behavioural and technical factors, Bettenburg et al. (2008) researched the characteristics of bug reports that were the most effective for developers. The authors found that testers varied considerably in the amount of detailed information they included in their bug reports, and this variation impacted the subsequent technical aspects of fixing the bugs. Considerable research has also focused on using behavioural factors to predict failures in software development. Based on the findings of this research, software development communities have developed testing environments for forecasting risky software features and planning upcoming maintenance activities (Wiklund et al., 2017). Developers have focused not only on testing factors, but also on practitioners' behaviours (e.g., interactions around software development features) to boost failure predictions, find solutions, and prevent software bugs (Dam et al., 2018).

Several studies have also examined the attributes of software developers on the Stack Overflow platform. For example, Bazelli et al. (2013) examined Stack Overflow question and answer (Q&A) threads to identify developers' characteristics using linguistic analysis and concluded (unsurprisingly) that the members with the highest reputation were more confident than those with average or below average reputation. Furthermore, members who used the up-voting feature of the platform were considerably less negative overall than those who used the down-voting feature. In addition, studies by Calefato et al. (2015) and Novielli et al. (2014) examined the Stack Overflow platform to determine developer characteristics that increased the likelihood of a technical contribution receiving a satisfactory answer. The authors indicated that factors linked to information exposition, time, and the emotional markers found within the thread influenced the possibility of getting a successful answer. Overall, the findings suggest a link between the behaviour of developers, developers' accomplishments, and their contributions to questions and answers posted on Stack Overflow.

Other studies have focused on the role of Stack Overflow contributors' geographical location. Schenk and Lungu (2013) found that Stack Overflow contributors from Europe and North America dominated the contribution of content while users from Asia consumed far more information from the platform than they contributed. This study also found that the individuals creating the most content also readily identified their location (Schenk and Lungu, 2013). Nivala et al. (2020) found that participation on Stack Overflow is significantly influenced by geographical location and local conditions. Oliveira et al. (2018) found that contributors from Western countries are far more likely to contribute to Stack Overflow than those from Eastern countries. Oliveira et al. (2016) found that the percentage of contributors who contribute to Stack Overflow varies widely per country and that this variation can be explained by both proficiency in English and culture.

Based on previous research, we expect the attitudes of developers from individualistic cultures to differ from those of developers from collectivistic cultures on Stack Overflow. An attitude is defined as a stable way of conceiving or sensing someone or something that is commonly mirrored in an individual's behaviour (Stevenson, 2010). Members of collectivistic cultures exhibit a strong disposition towards prioritising community objectives over personal goals (Han, 2004). Also, they gravitate towards group reasoning to avoid disruptions and accomplish community objectives (Choi et al., 2016), preferring to maintain group traditions and norms. Individuals from collectivistic cultures tend to be more interactive and to comply with group settings and goals more favourably than those from individualistic cultures (Elleson, 1983; Mann, 1980). They also tend to prefer

indirect procedures to resolve conflict instead of straightforward formal resolutions (by highlighting contextual importance over the content) and to favour indirect communications more than direct ones (Gudykunst, 1997). Additionally, Rice et al. (1998) found that members of collectivistic cultures prefer to observe reactions when they are in an asynchronous medium in order to maintain a cohesive group environment. We aim to explore such differences among Stack Overflow contributors by answering RQ2: *Do the attitudes of software development practitioners, on Stack Overflow, from collectivistic cultures differ from those who come from individualistic cultures? If so, what is the magnitude of these differences?* Developers from individualistic cultures may be more self-focused on Stack Overflow, while developers from collectivistic cultures may be more likely to express agreement with other members. Such patterns could have implications for group work, team cohesion, and knowledge sharing more generally.

Individuals' interactions and engagement in communication channels are crucial factors for knowledge production and distribution during crowd work (Van Den Hooff and De Ridder, 2004). Crowd work is defined as collaborative group work on a set of tasks to enhance members' learning capabilities and improve outputs. It involves knowledge contribution and knowledge acquisition, both of which are influenced by intellectual and inspirational determinants (Van Den Hooff and De Ridder, 2004). Sowe et al. (2008) utilised the mailing list records of the Debian project to inspect knowledge sharing between developers. They noticed that no one person controlled knowledge sharing operations, suggesting that knowledge sharing was dispersed widely without centralised control. In related research, Abreu and Premraj (2009) investigated the mailing list records of the Eclipse project and observed that communication strengthened as the number of introduced bug changes increased. They further discovered that developers interacted with each other the most often during delivery and integration periods, suggesting that the frequency of interaction was positively correlated with the need to share knowledge to meet the technical demands of the project.

Knowledge sharing is the process of capturing, storing, sharing, and using knowledge (Ouriques et al., 2019). In the current study, knowledge sharing is represented by various communication channels that are affected by individuals' willingness to share knowledge with others (Rashid et al., 2019). Prior research has revealed that interaction is a crucial factor influencing the success of software development activities (Hall et al., 2007). Energetic engagement of group members positively influences group processes and allows the community to experience a singular vision (Graziotin et al., 2018) that increases the probability of the community successfully accomplishing its goals (Herbsleb and Moitra, 2001). Conversely, negative behaviours can damage community balance and coherence in software development environments (Chang et al., 2013), ultimately having a negative impact on community functioning (Graziotin et al., 2018). For example, Ardichvili et al. (2006) investigated the cultural factors that affect knowledge sharing strategies within an online community and found that members of the community from different countries demonstrated varying levels of interest in participating in information exchange activities due to competition amongst themselves. Such behaviours during software development or among software developers operating in group settings could have a significant negative impact on project outcome and success.

Consistent with previous research, this study examines cultural factors that influence interaction and knowledge sharing in software engineering using Hofstede's individualism versus collectivism cultural dimension. We speculate that in a collaborative development context, members from a collectivistic culture are likely to have a higher behavioural tendency towards interaction and knowledge sharing than members from an individualistic culture. Investigating the individualism dimension within the context of software engineering will validate Hofstede's findings regarding business management in a software development setting. It will also inform interactions between cultures and collaboration strategies of developers operating in individualistic and collectivistic cultures. Therefore, RQ3: *How does the process of interaction and knowledge sharing compare between software development practitioners, on Stack Overflow, from collectivistic cultures versus those from individualistic cultures?* was developed to explore the process of interaction and knowledge sharing and how it varies among countries.

3. Methods

3.1. Country selection

Software development practitioners on Stack Overflow from three countries – the United States (US), China, and Russia – were selected for this study. Multiple reasons guided the selection of these three countries. First, they are considered superpowers in their respective regions (Jorgenson and Vu, 2005). Although Russia's progress on harnessing the internet for economic activity has not been as rapid as that of the US or China, it is an interesting example of a high-income country with a moderate internet technology pace (International Telecommunication Union, 2017) and a useful benchmark for comparison against the US. Second, all three countries are advanced and highly developed countries, with a high proportion of internet users.[2] We propose that the stability in the high proportion of internet users is also accompanied by rapid technological development, which can illustrate how culture has blended into each country's software engineering industry. Third, these three countries were selected based on their ability to make the internet inclusive for all. Finally, the three countries cover a broad range of scores on the individualism–collectivism dimension of Hofstede's framework, ranging from low (China) to moderate (Russia) to high (the US). An additional factor for potential consideration is Russia's unique position as straddling Western and Eastern culture. Its unique culture draws from multiple traditions and thus serves as an interesting point of comparison between Western (US) and Asian (China) culture.

Contributors' countries of origin were detected by scanning for mentions of locations in their profiles using the Python library *Geotext*.[3] Although the tool is not able to detect the geographical location of all contributors due to a number of limitations (e.g., lack of profile, no mention of geographical location), the geographical spread it detects reflects the results of the 2019 Stack Overflow survey (Bachschi et al., 2020).

3.2. Sample

Selecting a rich and varied sample enables researchers to draw relevant conclusions that comprehensively address the studied phenomenon (Aaltio and Heilmann, 2010). In this study, expert sampling approaches were employed to (1) select cases that provided detailed information on software engineering culture; (2) increase accuracy and eliminate bias from the obtained data (Yin, 2009); and (3) help mitigate hardware-related limitations. Therefore, the data for this study were archival data from Stack Overflow, spanning more than 11 years (September 2008 to September 2019). Although there are multiple data sources

[2] https://data.worldbank.org/indicator/IT.NET.USER.ZS.

[3] https://github.com/elyase/geotext.

available on Stack Exchange, only the data from Stack Overflow were analysed from the Stack Exchange superset.[4]

From the initial source, we extracted all entries that contain part of the following strings through basic SQL queries: "CHINA, China, china, RUSSIA, Russia, russia, UNITED STATES, United States, united states, USA, usa and U.S". We then queried entries belonging to each specific location and inserted these records into a new table. Therefore, we ended up with three newly created tables where '32,974' users who included China in their location field were inserted into a new table named Users_China, '14,349' users from Russia (Users_Russia) and '82,192' users from the United States (Users_USA). It should be noted that profiles with multiple location references were excluded from this extraction phase. Additionally, some users have mentioned only city names without a country reference such as "New Orleans" or stated an overlapping location name such as "America", which might also indicate a continent. For example, a user might suggest his/her location from 'Brazil, South America' while another user might state their location 'Detroit, America' or 'Detroit, Michigan, America'. Therefore, to enhance the robustness of our method, we decided to use GeoText to include entries where our initial basic SQL queries could not distinguish such entries. The GeoText tool has the ability to automatically detect countries from city references if such references are unique and exists within the GeoNames database,[5] otherwise the tool assigns a 'null' value if such references cannot be detected successfully. The tool was able to identify the location of '10,669' additional entries, including '4702' entries belonging to the United States, '2105' entries from China and '3862' entries from Russia. Combining results from both methods (SQL queries and GeoText), we were able to identify a total of 140,184 users, including 86,894 users from the United States, 35,079 users from China, and 18,211 users from Russia.

To assess the GeoText tool's reliability, a random sample of 384 contributor records were selected from the original dataset and analysed manually and by the GeoText tool to obtain a 95% confidence level and a 5% error margin (Faul et al., 2007). Considering the manual check to be the 'ground truth', the tool-processed entries were cross checked against the manual ones which resulted in 315 matched countries and a reliability of 82.03%. That said, for many of the cases that did not matched, GeoText did not actually recommend a country, instead assigning "null". Thus, such records would not be included in our study, as we only studied records where United States, China or Russia were clearly discernible. Our results thus may have missed some records, but we are confident that the users studied for these countries indeed self-identified those as their country of origin. This detail is now added to Section 6. The number of records by database table is displayed in Table 1 for these three countries. Our data are publicly available here: https://console.cloud.google.com/marketplace/details/stack-exchange/stack-overflow.

Beyond the three countries studied, altogether, 141 countries are represented in the dataset, including profiles from North America like Canada with 80,582 users, profiles from South America like Chile with 7803, Colombia with 12,425, Costa Rica with 2091 and Brazil with 86,073 users. The dataset also includes profiles from the European continent, including United Kingdom with 135,890 users and Sweden with 27,414 users, profiles from the African continent like Kenya with 12,434 users and Nigeria with 26,215 users, profiles from the Asian continent including Malaysia with 18,472 users and Thailand with 14,369 users. Finally, the dataset has profiles from the South Pacific like Australia with 58,553 users and New Zealand with 11,508 users. In total, 3,255,955 contributors identified their location, which is 30.9% when considering the 10,532,295 total contributors in the dataset.

Table 1
Stack Overflow total records by database table by country.

Table	US	China	Russia
Users	86,894	35,079	18,211
Badges	532,143	107,418	93,647
Comments	2,733,789	186,361	158,392
Posts	1,697,153	169,333	226,447
Questions	791,310	78,062	117,805
Answers	905,843	91,271	108,642
Post history	1,339,700	254,088	100,995
Tags	1,947	113	96
Votes	84,830	158,089	84,830

3.3. Measures

Orientation (RQ1): We developed measures to assess the type of orientation software practitioners from collectivistic and individualistic cultures exhibit when operating in the community. Orientation was defined as the way software practitioners contribute their social and intellectual capacity to the software development community (Li et al., 2001). Developers' orientation was measured using their artefacts from the Stack Overflow platform. We sought to extract all the quantitative footprints that display developers' contribution activities on the Stack Overflow platform. These dimensions are central to the Stack Overflow platform's gamification mechanism, and were of utility to our study goals, especially since this study focuses on individuals' interdependence and collaboration". We extracted 11 quantitative dimensions of orientation mainly from the *Users* table, with join operations between the *Users* table and other tables from Stack Overflow — based on their unique mutual identifiers providing additional orientations. For instance, we use the *Posts* table to differentiate whether posts added by users are questions or answers. These records were all combined to form one new holistic table named *Orientation*. These join operations enabled us to identify users' specific quantitative contributions on the platform (e.g., how many questions were asked, how many answers and comments were provided, and so on).

Given our drive to include all the possible measures, we anticipate that our work will be all encompassing, however, previous work have examined several of these dimensions individually (Bachschi et al., 2020; Tian et al., 2019). In addition, our quantitative measures were triangulated by deeper contextual approaches, where in the first phase of the work we wanted to analyse the data as it is, and not modify these through artificial means. These quantitative variables in turn enabled us to identify how users were rewarded for their efforts on Stack Overflow, such as how many badges were earned, and the reputation score awarded to each user. The 11 orientation dimensions are further explained in Table 2. A screenshot highlighting some of the orientation dimensions for one Stack Overflow developer is presented in Fig. 1.

Attitudes (RQ2): We employed the same five linguistic dimensions used in previous research (Licorish, 2013; Licorish and MacDonell, 2014b) that analysed software developers' textual information to understand their personality, attitude, and emotion patterns. Beyond this work, these linguistic categories were also assessed in other works (e.g., Licorish and MacDonell (2014a) and Licorish and MacDonell (2013c)). These five dimensions were extracted from contributors' "About Me" profiles: (1) pronouns, (2) cognitive, (3) work and achievement, (4) leisure, social, and positive emotion, and (5) negative emotion. For the first dimension, first-person singular pronouns such as "I" are indicators of individualistic and self-focused attitudes (Pennebaker and Lay, 2002), while plural pronouns such as "we" are indicators of

[4] https://archive.org/details/stackexchange.
[5] http://www.geonames.org/.

Table 2
Descriptions of developers' orientation dimensions.

Orientation dimension	Description
AboutMe length	The number of words developers wrote about themselves in their "About Me" profile.
Duration on site (months)	The number of months a user used the Stack Overflow platform, which was calculated as the last access date minus the profile creation date.
Up votes	A measure assigned by the community to reflect the usefulness of questions and answers posted by a user. Users issue up votes by clicking an up arrow to the left of a post or comment.
Down votes	A measure assigned by the community to indicate posted questions and answers offering little value. Users issue down votes by clicking a down arrow to the left of a post.
Reputation	A measure calculated by the Stack Overflow platform to reflect the amount of trust the community has in a user. A user gains reputation when a question or answer is voted up (+10), an answer is marked accepted (+15), or a suggested edit is accepted (+2). They lose reputation when an answer is voted down (−2 for the user, −1 for the voter). A user can gain at most 200 reputation points per day from these activities.
Views	The total number of times a user profile has been viewed.
Badges	The total number of badges earned by a user. Badges are awarded based on a user's contribution to the site when asking or answering questions, moderating an activity, or participating in non-community-wiki questions (tag-based). There are three categories of badge: gold, silver, and bronze.
Comments	The total number of comments provided by a user. A user can post a comment to discuss a question or answer and ask for further clarification without the need to post a new thread. The length of a comment is restricted to 15–600 characters.
P questions	The total number of questions asked by a user.
P answers	The total number of answers provided by a user.
Post history	The total number of times a user has edited posts (questions or answers) to improve them.

collectivistic attitudes and a shared group focus (Pennebaker and Lay, 2002). The second-person pronoun "you" is an indicator of the degree of members' reliance on one another and a collectivistic attitude (Pennebaker and Lay, 2002). The second dimension represented the intellectual and behavioural aspects of developers. Previous research has established that software communities are most productive when their members have high cognitive capabilities, which makes them good problem solvers (André et al., 2011). The third dimension of work and achievement-related words indicated individualistic attitudes and attitudes favouring task completion (Benne and Sheats, 1948). Conversely, the fourth dimension of leisure, social, and positive emotion indicated collectivistic behaviours and the promotion of collectivistic behaviour in others (Van Den Hooff and De Ridder, 2004). Research suggests individuals with collectivistic attitudes tend to be social and use positive and group-related words in their communications (Triandis, 1998). The fifth dimension of negative emotion indicated members who contribute negatively to the behavioural climate of groups. Individuals who express negative emotions usually reflect unfulfillment, dissatisfaction, and individualistic behaviours that can impact group cohesiveness (Denning, 2013). These dimensions have been used previously (e.g., Coltheart (1981), Gill and Oberlander (2003), Iyer (2019), Licorish and MacDonell (2013b) and Mairesse et al. (2007)) to study the attitudes that were expressed by Stack Overflow users from the three countries considered (the US, China, and Russia). Table 3 provides a summary of the linguistic dimensions that were considered, with examples.

Interaction and knowledge sharing (RQ3): This qualitative phase of the study used a bottom-up content analysis (CA) classification protocol created by Licorish and MacDonell (2014b) that has 13 encoding categories (see Table 4) that were validated using multiple preliminary trials in order to explore the relationship between culture and software developers' interaction and knowledge sharing patterns (Krippendorff, 2018). Such data-driven approaches generate the dimensions for classification from the data rather than from predefined dictionaries. Consequently, bottom-up techniques are believed to grasp the precise context where phrases are utilised (Damerau, 1993). This particular coding schema has previously been used to explore engagement and knowledge sharing, resulting in ground-breaking findings in computer group learning and interactions (Zhu, 1996), and more recently in the assessment of the patterns of interactions among genders (Zolduoarrati and Licorish, 2021). Licorish (2013) found that software engineers communicate several ideas in their messages. Therefore, developers' messages were segmented into sentences, and these sentences were the primary analysis units in this study.

3.4. Data analysis

We performed data transformation by normalising the data, dividing by the totals to adjust for an imbalanced dataset. For example, the total number of questions asked for a country was divided by the total number of users of that country. . Normalising the data enabled us to ensure the results for developers who have been on the platform longer (i.e., giving them more time to ask questions and provide answers) did not have a disproportionate impact on the results, since US and Russian developers were typically on the platform for longer than Chinese developers.

For the orientation metrics (RQ1), we used SPSS to conduct exploratory data analysis and statistical analyses (Tukey, 1977). The main objective of the exploratory analysis was to understand developers' orientation in relation to their culture (i.e., collectivistic or individualistic). After conducting the exploratory analysis, we tested for the independence of the three samples. We believed the samples were independent because the developers from the three countries were geographically separate from each other without overlap. Levene's test for homogeneity of variance (Levene, 1961) and the Brown–Forsythe test, which is a more robust variation of Levene's test (Brown and Forsythe, 1974), were used to test whether the variances in the orientation metrics were equal across the three samples (Mangiafico, 2016). Next, we performed a Kolmogorov–Smirnov normality test (Öztuna et al., 2006) test to check whether the orientation metrics from each country were normally distributed. If so, then parametric tests were used; otherwise, non-parametric tests were used (Kaur and Kumar, 2015). We also tested whether there was a true difference in the developers' orientations across the three countries. If the data required non-parametric treatment, the Kruskal–Wallis test was the preferred procedure for comparing more than two independent samples (Vargha and Delaney, 1998). To further check for differences between any two of the three countries' non-parametric data, we utilised a Mann–Whitney test (Nachar, 2008).

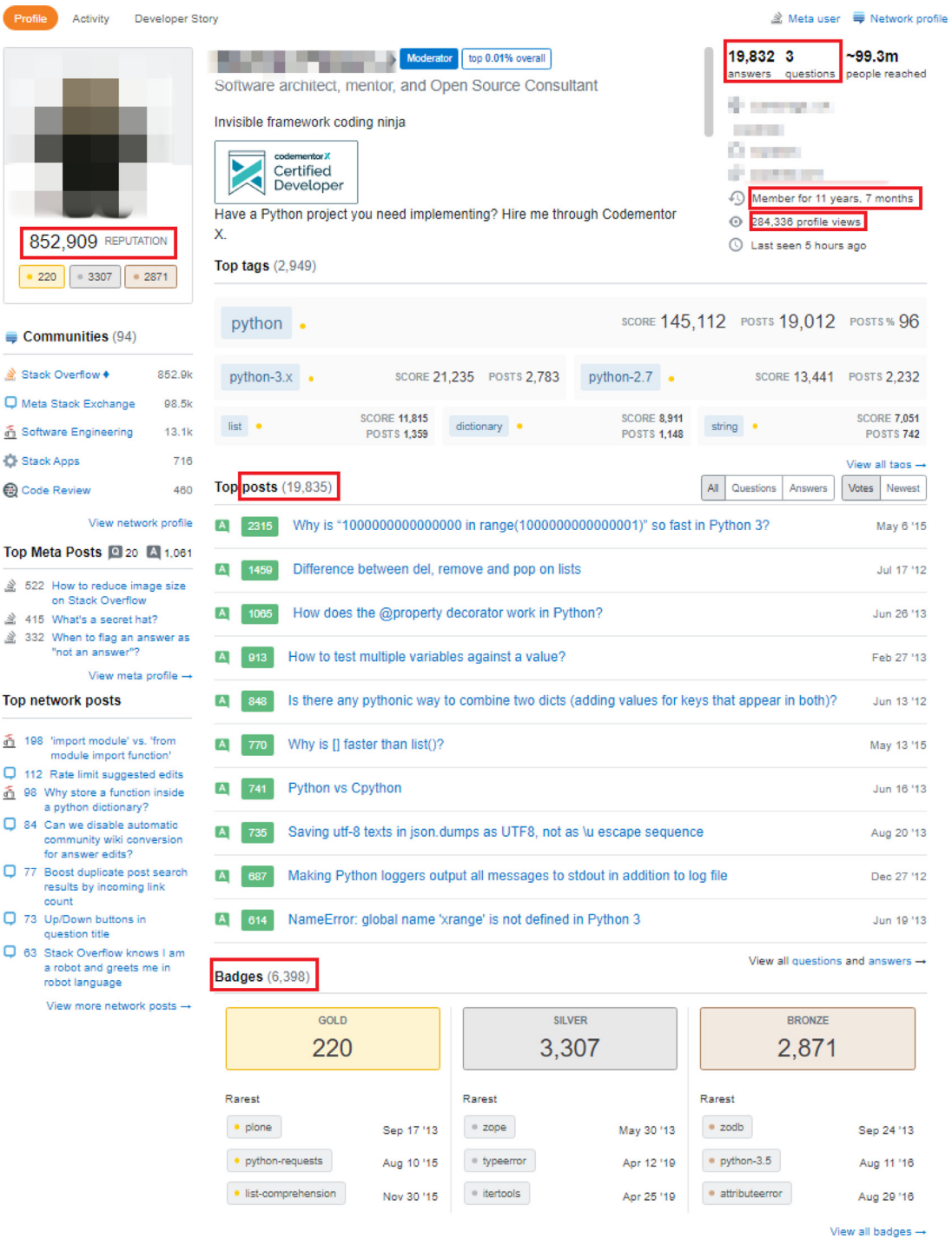

Fig. 1. Screenshot highlighting major orientation dimensions (Reputation, P questions, P answers, Duration on site, Views, and Badges) for one Stack Overflow developer.

All tests used a 5% significance level. Because it is possible to obtain significant results by chance (Type I error) when multiple statistical tests are done on the same dependent variable, we used a Bonferroni correction to compensate for this (Weisstein, 2004).

For developers' attitudes (RQ2), we utilised Linguistic Enquiry and Word Count (LIWC) software (Iyer, 2019) to analyse the textual information in the "About Me" profiles. LIWC is a text analysis engine based on 40 years of data accumulation in the US, New Zealand, and Canada (Tausczik and Pennebaker, 2010). The compiled knowledge used in LIWC dictionaries stretches over several genres, including supervised and sentimental writing, articles, regular conversations, blog entries, and stories. The tool recognises over 86% of conversational words and is composed of 4500 linguistic words accessible in several languages. Within the tool itself, word clusters are located within specified genres or categories, such as social words, positive emotion, negative emotion, quantifiers, and so on. Textual data is imported into the tool as an input file and is processed by matching every word within the document against LIWC dictionary words.

Table 3
Description of LIWC linguistic categories.

Linguistic dimension	Variable	Samples	Total words
Pronouns			
Singular pronouns (1st person)	I	I, me, my, mine	24
Plural pronouns (1st person)	We	We, us, our	12
2nd person pronouns	You	You, your, yours	30
Cognitive			
Insight	Insight	Think, know, consider	259
Discrepancy	Discrep	Should, would, could	76
Tentative	Tentat	Maybe, perhaps, guess	178
Certainty	Certain	Always, never, confident	113
Work and Achievement			
Work	Work	Job, majors, xerox	444
Achievement	Achieve	Earn, hero, win	186
Leisure, Social, and Emotions			
Leisure	Leisure	Cook, chat, movie	296
Social processes	Social	Talk, us, child, friend	756
Positive emotions	Posemo	Love, happy, pretty, nice	620
Negative emotions	Negemo	Hate, hurt, nasty, ugly	744

Table 4
Interaction and knowledge sharing coding categories.
Source: Adapted from Licorish (2013).

Scale	Category	Explanation
1	Type I Question	Requests solution or answer due to a knowledge deficit, e.g., "Which class contain the implementation and deployment for screen hibernation feature?"
2	Type II Question	Initiates discussion, e.g., "let us talk about the new python method that discards the unique index and has even weirder side effects in more complicated cases".
3	Answer	Provides feedback or response for knowledge seekers, e.g., "You can find hibernation features implemented across these classes (HMC1, CYMH, DHH and 3HC), I would suggest that you follow a similar approach".
4	Information exchange	Shares knowledge, e.g., "You do not need to do that since the API team were able to crack down issue number 315 yesterday".
5	Discussion	Provides an elaboration that expresses ideas or thoughts, e.g., "Solving issue #138 helped to solve the error produced in the (field_automation) class since it took care of all refactoring problems".
6	Comment	Provides statements, e.g., "I highly believe that test should be implemented first using a test-driven approach where tests fail at the start".
7	Reflection	Provides an appraisal or self-evaluation, e.g., "I have noticed that the MVC framework from last year project can be applied to the current one besides including the useful techniques learnt in that challenging project".
8	Scaffolding	Provides a proposition and advice, e.g., "I think it is a better idea to use clear and elaborative comments when we code to help in the final production of the documentation file".
9	Instruction/Command	Gives a directive, e.g., "Fix or delete your posted answer since your provided pattern is neither a mixture of regex nor like-clause".
10	Gratitude	Provides appreciation or praise, e.g., "Nice, your solution actually worked, thanks for the post".
11	Off Task	Provides an unrelated transmission of messages regarding the current task or post, e.g., "Admin it has been a while".
12	Apology	Expresses remorse, e.g., "I do apologise for posting a question with the wrong…"
13	Not coded	Communications that cannot be assigned to any of the preceding twelve categories.

The results are summarised by lexicon categories, which consist of matched word count generated by the lexicon (e.g., the standard psycholinguistic pronoun dimensions, the psychological genres, and functional phrases including the positive and negative sentiment categories as well as the social, leisure, and work categories). Specifically, every word within the "About Me" profiles was matched against LIWC dictionary words, and the results were summarised.

An alternative to LIWC is the Medical Research Council (MRC) Psycholinguistic Database. While both utilise established dictionaries in the text analysis process, research has found that LIWC is more accurate than the MRC database when assessing behaviours in communication (Mairesse and Walker, 2006). Moreover, LIWC has also found wider support in terms of being linked to attitudes in the psycholinguistic literature compared to the MRC database (Yee et al., 2011). Consequently, this study employed LIWC to examine software practitioners' attitudes based on the language expressed in their messages.

For developers' interaction and knowledge sharing patterns (RQ3), we used bottom-up content analysis (Krippendorff, 2018) with a directed classification protocol created by Licorish (2013), comprising 13 encoding categories (see Table 4). The original sample size of the communications in the Posts table was 1,697,153 for the US, 169,333 for China, and 226,447 for Russia. The G*Power statistical tool was used to identify an appropriate random sample size from the original population of posts to minimise the coding burden (Faul et al., 2007). To obtain a 95% confidence level and a 5% error margin, 384 posts were needed from each population. In terms of the distribution of posts sampled, we randomly selected 180 questions and 204 answers from the US, 178 questions and 206 answers from China, and 199 questions and 185 answers from Russia. When proportionally assessed against the overall distribution of questions and answers across the entire dataset in Table 1 (i.e., total questions/total answers), these samples are almost identical (i.e., US = 0.88 and 0.87, China = 0.86 and 0.86, and Russia = 1.08 and 1.08). After the extraction process, we copied records for each population to Microsoft Excel for easy management during the coding process. Two coding phases were conducted. In the first coding phase, the researchers (first and third authors) were the main coders and

categorised roughly 5% of the three countries' communications (20 messages from each country for 60 in total) for piloting and reliability checks. Encoding discrepancies were discussed and resolved through consensus. Cohen's Kappa coefficient indicated an agreement level between coders of 0.831 (Cohen, 1960), which is considered an acceptable level of inter-rater reliability. During the second coding phase, all messages from each sample were inspected individually to see if they captured interaction and knowledge sharing. If so, we segmented the message into sentences and coded each sentence according to the Licorish (2013) scale. At the start of the coding procedure, all 13 categories were assigned a zero mark. Subsequently, sentences that captured information and knowledge sharing were selected and matched against the coding schema. If a match was found, then a mark of one was assigned to the matched category. Otherwise, it remained zero. For sentences belonging to more than one category, a mark of one was assigned to both categories as recommended by content analysts (Zhu, 1996).

4. Results

4.1. Developers' orientation (RQ1)

To answer RQ1, the descriptive statistics for the 11 orientation dimensions including the mean (*M*), median (*Med*), standard deviation (*SD*), skewness (*SK*), and kurtosis (*KS*) are shown in Table 5. One notable finding is that US developers had higher means than Chinese and Russian developers across all orientations apart from "Comments" (US = 1.93, China = 5.72, Russia = 3.39), "Post history" (US = 14.7, China = 26.7, Russia = 1.68), and "Duration on site (months)" (US = 35.49, China = 24.85, Russia = 36.29).

Conversely, an inspection of the medians revealed that approximately half of the developers' orientation dimensions had a median of zero, because more than half the developers did not complete the "About Me" section (the total number of users with no "About Me" orientation was US = 60,774, China = 18,509, Russia = 11,460). Similarly, median values for other orientations were very close to zero because a large proportion of the developers simply did not use that specific orientation feature of the Stack Overflow platform.

The mean was not equal to the median across the 11 orientations, indicating the orientations were not normally distributed. Additionally, it is discernible that "Reputation" orientation has the largest mean across all 11 orientations, which indicate that users from all three countries are well-reputed and receive up votes particularly frequently.

Furthermore, inspecting skewness in detail, it is apparent that "Duration on site (months)" orientation were close to zero across all three countries, indicating that most developers use the platform for roughly the same duration, regardless of their origin country (the skewness of "Duration on site (months)" orientation was US = 0.74, China = 1.25, Russia = 0.85)

After examining the descriptive statistics, we performed the Kruskal–Wallis statistical test (see Table 6) to examine the outcomes for significant differences. The results suggest there were significant statistical differences in the orientation of developers across the three countries ($p < 0.01$).

Explaining these outcomes, the statistics in Table 5 showed that US developers had a longer "About Me" length (US = 88.33, Russia = 44.83, China = 26.15), received more "Up votes" (US = 67.44, Russia = 56.82, China = 15.14) and "Down votes" (US = 8.44, Russia = 3.60, China = 1.17), had a higher "Reputation" (US = 722.08, Russia = 409.21, China = 142.51), received more profile "Views" (US = 86.92, Russia = 44.75, China = 18.13) were awarded more "Badges" (US = 10.96, Russia = 7.51, China = 3.80), and posted more questions (US = 4.78, Russia = 2.89, China = 1.70) and answers (US = 14.70, Russia = 9.50, China = 3.11) than Chinese and Russian developers.

Table 5
Descriptive statistics for developers' orientation dimensions across the three countries.

Orientation	Country	*M*	*Med*	*SD*	*SK*	*KS*
AboutMe length	US	88.33	0.00	231.47	6.02	55.82
	China	26.15	0.00	103.54	12.18	238.98
	Russia	44.83	0.00	161.24	10.09	179.53
Duration on site (months)	US	35.49	25.82	33.90	0.74	−0.56
	China	24.85	11.65	29.43	1.25	0.58
	Russia	36.29	22.21	37.83	0.85	−0.44
Up votes	US	67.44	0.00	399.89	23.79	1 157.37
	China	15.14	0.00	132.30	27.49	1 170.13
	Russia	56.82	0.00	415.02	41.36	2 763.30
Down votes	US	8.44	0.00	239.57	112.85	17 546.13
	China	1.17	0.00	34.30	86.23	10 358.91
	Russia	3.60	0.00	56.01	48.46	3 265.98
Reputation	US	722.08	6.00	7949.08	53.95	4 394.94
	China	142.51	1.00	2694.45	116.82	17 190.93
	Russia	409.21	1.00	4796.66	45.37	2 608.95
Views	US	86.92	3.00	1517.16	119.02	18 360.44
	China	18.13	0.00	319.13	88.09	9 295.42
	Russia	44.75	2.00	422.52	45.77	2 847.42
Badges	US	10.96	2.00	40.85	31.41	2 198.05
	China	3.80	1.00	17.12	50.68	5 030.93
	Russia	7.51	2.00	27.10	30.35	1 650.19
Comments	US	1.93	0.00	15.16	72.84	7 480.36
	China	5.72	0.00	57.97	54.19	4 474.58
	Russia	3.39	0.00	28.30	71.96	6 361.21
P questions	US	4.78	0.00	20.90	21.03	915.13
	China	1.70	0.00	9.81	19.66	580.53
	Russia	2.89	0.00	15.77	32.10	1 692.53
P answers	US	14.70	0.00	254.46	159.43	34 315.92
	China	3.11	0.00	38.22	68.00	6 116.01
	Russia	9.50	0.00	94.26	42.21	2 355.58
Post history	US	14.70	0.00	187.37	67.36	6 362.80
	China	26.70	0.00	227.51	55.46	3 912.10
	Russia	1.68	0.00	26.49	56.57	4 057.27

M = mean, *Med* = median, *SD* = standard deviation, *SK* = skewness, *KS* = kurtosis.

Conversely, Chinese developers provided more "Comments" (China = 5.72, Russia = 3.39, US = 1.93) and edited more posts ("Post history"; China = 26.70, US = 14.70, Russia = 1.68) than US and Russian developers.

Finally, Russian developers had used the site longer (Russia = 36.29, US = 35.49, China = 24.85) than US and Chinese developers and also provided more "Comments" than US developers, as noted above.

We examined the mean ranks to see if these outcomes still held given the skewness of the data. Our outcomes in Table 6 revealed identical patterns to those found in Table 5, with US developers ranking first on most orientation metrics apart from "Comments" and "Post history" (both led by China), and "Duration on site (months)" (led by Russia). Mann–Whitney tests were performed to identify differences in the three pairs of countries, taking into account Bonferroni corrections, and the results revealed all pairwise differences were also statistically significant ($p < 0.01$).

4.2. Developers' attitudes (RQ2)

To answer RQ2, the descriptive statistics for the linguistic scores of developers are displayed in Table 7 for all variables across all three countries. For the pronouns dimension, the words

Table 6
Kruskal–Wallis test results for developers' orientation dimensions across the three countries.

Orientation	Country	Mean rank	Chi-square	*df*	Asymp. *p*
AboutMe length	US	74 624.77			
	China	61 708.01	4 010.356	2	<0.01
	Russia	64 617.38			
Duration on site (months)	US	62 070.71			
	China	45 613.15	1 949.087	2	<0.01
	Russia	72 173.89			
Up votes	US	73 666.25			
	China	60 221.06	4 243.140	2	<0.01
	Russia	72 055.18			
Down votes	US	71 981.17			
	China	65 022.74	2 393.490	2	<0.01
	Russia	70 846.31			
Reputation	US	75 208.03			
	China	57 039.64	5 911.730	2	<0.01
	Russia	70 826.82			
Views	US	76 176.04			
	China	54 544.38	7 592.133	2	<0.01
	Russia	71 014.41			
Badges	US	76 147.24			
	China	55 275.66	6 877.029	2	<0.01
	Russia	69 743.23			
Comments	US	56 534.48			
	China	77 171.98	13 206.411	2	<0.01
	Russia	59 594.51			
P questions	US	75 730.91			
	China	58 530.84	5 479.561	2	<0.01
	Russia	67 177.20			
P answers	US	73 497.50			
	China	60 596.44	3 425.781	2	<0.01
	Russia	72 137.34			
Post history	US	55 591.25			
	China	80 093.23	19 839.151	2	<0.01
	Russia	50 306.98			

"I" and "You" were used more frequently than the word "We", as indicated by the reported mean and median measures.

Also, US developers were more individualistic and used the pronoun "I" more frequently ($M = 2.10$) than Russian ($M = 1.92$) and Chinese ($M = 1.75$) developers. Conversely, Chinese and Russian developers were more collectivistic and used the pronouns "You" (China = 1.96, Russia = 1.68, US = 1.48) and "We" (China = 0.13, Russia = 0.12, US = 0.11) more frequently.

For the cognitive dimension, Chinese developers used insightful and tentative words ($M = 1.72$ and $M = 4.03$) more frequently than Russian ($M = 1.59$ and $M = 2.54$) and US ($M = 1.56$ and $M = 2.18$) developers. Conversely, US developers used discrepancy and certainty words ($M = 1.80$ and $M = 1.47$) more frequently than Chinese ($M = 1.55$ and $M = 0.72$) and Russian ($M = 0.63$ and $M = 0.90$) developers.

For the work and achievement dimension, US developers ($M = 2.38$ and $M = 1.92$) had higher mean scores than Russian ($M = 2.35$ and $M = 1.26$) and Chinese ($M = 1.52$ and $M = 1.21$) developers. For the leisure, social, and emotions dimension, Chinese developers used leisure, social, and positive emotion words ($M = 0.82$, $M = 4.38$, $M = 2.46$) more frequently than Russian ($M = 0.36$, $M = 3.80$, $M = 1.78$) and US ($M = 0.36$, $M = 1.27$, $M = 1.75$) developers, whereas words conveying negative emotions were used more frequently by US developers ($M = 0.62$) than by their Chinese ($M = 0.55$) and Russian ($M = 0.37$) counterparts.

In summary, the descriptive statistics suggested there were differences in both the means and medians among developers from the three countries. Given the sample sizes and the normality violations of the data indicated by the Kolmogorov–Smirnov test, we used a Kruskal–Wallis test to further examine statistically significant differences among developers' attitudes from the three countries. Outcomes are provided in Table 8. The test again revealed identical patterns to those found in Table 7, with the exception that Russian developers (1 003 669.93) used leisure words more frequently than US developers (980 926.09), compared to their equal ranking in Table 7. All of these results were statistically significant ($p < 0.01$). The pairwise differences were analysed using Mann–Whitney tests, which also revealed significant differences between all pairwise combinations of the three countries ($p < 0.01$).

4.3. Developers' interaction and knowledge sharing patterns (RQ3)

This section presents the results for RQ3, which examines the interaction and knowledge sharing patterns among developers from individualistic and collectivistic cultures. We used directed content analysis to categorise interaction and knowledge sharing behaviours among developers from the US, China, and Russia (see Table 4). Developers' posts were coded and transformed into numbers, which were analysed using frequency counts and percentages. We also sought to triangulate the results with those of the quantitative analysis of orientation and attitudes presented earlier. A chi-square association test was used to investigate the association between two or more samples (Kiliç, 2016). The test determines whether there is a significant association among samples or no association among samples. The test was applied to the coded software developer interactions.

Figs. 2 and 3 provide aggregated summaries of developers' interaction and knowledge sharing patterns. These figures reflect the outcomes of our content analysis, where, to obtain a 95% confidence level and a 5% error margin, 384 posts were selected from each population (i.e., all three samples were the same). Given that the samples used for this analysis were identical, there was no need to normalise the outcomes in these figures. Fig. 2 shows the total frequency count in each category across all three countries. This enables us to compare countries to show which country had the highest or lowest number of instances for a given category. Fig. 3 shows the percentages in each category for each country, which enables us to determine the most and least common categories of behaviour within each country. The behaviours were coded according to the 13 categories listed in Table 4.

As seen in Fig. 2, US developers were the most likely to ask **type 1 (direct) questions**. US developers asked a total of 156 direct questions, Russian developers came second with 129 direct questions, and Chinese developers third with only 27 direct questions. Although US developers were more likely to ask direct questions than Russian and Chinese developers, asking direct questions was not the most dominant category of behaviour for US developers. Additionally, as seen in Fig. 3, type 1 questions ranked fifth among US developers at 8.51% of interaction and knowledge sharing behaviours. Similarly, type 1 questions ranked fifth among Russian developers at 7.68%. Conversely, for Chinese developers, type 1 questions ranked tenth at 1.27%. Overall, the results suggest that US and Russian developers were similar in terms of their willingness to ask direct questions, although the US was slightly more dominant. However, Chinese developers were far less willing to ask direct questions than their US and Russian counterparts. We performed a chi-square test to examine these results for statistically significant differences, and it was observed that the results for the Type 1 Question category were significantly different: $X^2 = 109.28$, $df = 8$, $p < 0.001$.

Table 7
Descriptive statistics for developers' linguistic scores across three countries.

Category	Variable	Country	*M*	Med.	*SD*	SK	KS
Pronouns	I	US	2.10	1.27	2.58	1.68	4.20
		China	1.75	1.61	1.54	0.55	2.33
		Russia	1.92	1.73	1.82	0.92	2.86
	We	US	0.11	0.00	0.50	6.96	69.89
		China	0.13	0.00	0.53	6.78	69.02
		Russia	0.12	0.00	0.53	7.32	81.43
	You	US	1.48	0.27	2.21	2.16	6.63
		China	1.96	1.30	2.30	1.54	3.46
		Russia	1.68	0.91	2.18	1.81	4.77
Cognitive	Insight	US	1.56	1.04	1.83	2.20	8.92
		China	1.72	1.18	2.20	2.68	14.3
		Russia	1.59	1.13	1.92	2.09	8.45
	Discrep (= discrepancy)	US	1.80	1.47	1.78	1.56	14.14
		China	1.55	1.14	1.77	1.88	6.13
		Russia	0.63	0.00	1.20	3.54	21.73
	Tentat (= tentative)	US	2.18	1.71	2.24	1.83	4.41
		China	4.03	3.85	1.24	1.75	8.67
		Russia	2.54	2.33	2.27	1.52	6.72
	Certain (= certainty)	US	1.47	0.93	1.92	2.41	10.35
		China	0.72	0.00	1.23	3.34	21.33
		Russia	0.90	0.50	1.27	2.86	19.48
Work and Achievement	Work	US	2.38	1.78	2.57	2.18	8.38
		China	1.52	1.14	1.71	1.75	5.03
		Russia	2.35	1.32	2.17	2.05	6.76
	Achievement	US	1.92	1.89	1.76	2.61	3.82
		China	1.21	0.72	1.55	2.19	2.27
		Russia	1.26	0.81	1.62	2.43	3.13
Leisure, Social, and Emotions	Leisure	US	0.36	0.00	1.11	5.93	57.79
		China	0.82	0.00	1.30	3.03	17.62
		Russia	0.36	0.00	1.19	6.12	59.82
	Social	US	1.27	0.80	1.69	2.48	11.68
		China	4.38	3.77	3.34	1.30	3.17
		Russia	3.80	3.04	3.48	1.58	4.40
	Posemo (= positive emotion)	US	1.75	1.33	1.96	2.21	10.97
		China	2.46	1.75	2.79	2.05	7.69
		Russia	1.78	1.20	2.26	2.63	13.79
	Negemo (= negative emotion)	US	0.62	0.00	1.21	3.61	23.05
		China	0.55	0.00	1.04	3.71	25.49
		Russia	0.37	0.00	1.23	6.03	55.01

Similarly, US developers were more dominant than Russian and Chinese developers in providing **answers** to direct questions, providing a total of 363 answers. Russian developers came second with 109 answers, and Chinese developers third with just 12 answers. Among US developers, answers ranked second among the 13 behavioural categories at 19.8%. In contrast, answers ranked twelfth (second lowest) among Chinese developers at 0.57%, while Russian developers had answers ranked seventh at 6.49%. A chi-square test for the Answers category revealed statistically significant differences: $X^2 = 266.21$, $df = 20$, $p < 0.001$.

US developers were also more dominant in the **instruction** category. They provided instructions 187 times, their fourth-ranked behaviour (10.20%). Instruction ranked sixth (6.61%) among Russian developers, who provided a total of 111 instructions. Instruction was used the least by Chinese developers (56 instructions), for whom it ranked eighth (2.64%). The chi-square results for the Instruction category revealed statistically significant differences: $X^2 = 80.7$, $df = 14$, $p < 0.001$.

The final category that US developers led in was in **off task** category, which encompassed communication unrelated to the current task or post. US developers had the highest frequency for this category (40 interactions), but it was used relatively infrequently among US developers and ranked ninth (2.18%) among the 13 behavioural categories. Russian developers had the second highest frequency (31 interactions) and also used off task behaviours relatively infrequently, ranking tenth (1.85%). For Chinese developers, the off task category was ranked the lowest (0.57%) of all interaction and knowledge sharing categories and used the least frequently (12 interactions). The chi-square results for the Off Task category revealed statistically significant differences: $X^2 = 18.84$, $df = 10$, $p < 0.001$.

Conversely, Chinese developers led US and Russian developers in the asking of **type 2 (indirect) questions**. Chinese developers asked a total of 99 indirect questions, which was more than the number of indirect questions asked by Russian (75) and US (14) developers. Among Chinese developers, indirect questions ranked sixth (4.67%) among their interaction and knowledge sharing behaviours. In contrast, type 2 questions ranked thirteenth (the lowest, at 0.76%) among US developers and eighth (4.47%) among Russian developers. Indirect questions ranked lower than direct questions among Russian developers (eighth and fifth, respectively), reflecting the moderate preference of Russian developers for direct questioning. The chi-square results for the Type 2 Questions category revealed statistically significant differences: $X^2 = 62.97$, $df = 6$, $p < 0.001$.

Chinese developers were also more willing than US and Russian developers to share knowledge with others via **information exchanges**. Chinese developers participated in a total of 235

Table 8
Kruskal–Wallis test results for developers' linguistics scores across the three countries.

Category	Variable	Country	Mean Rank	Chi-square	*df*	Asymp. *p*
Pronouns	I	US	1 071 944.85	3 387.62	2	<0.01
		China	970 339.45			
		Russia	1 036 591.72			
	We	US	1 016 062.30	972.62	2	<0.01
		China	1 035 829.77			
		Russia	1 020 954.03			
	You	US	893 919.75	14 256.49	2	<0.01
		China	1 054 179.08			
		Russia	973 189.06			
Cognitive	Insight	US	1 002 186.70	507.47	2	<0.01
		China	1 035 808.46			
		Russia	1 030 884.23			
	Discrep (= discrepancy)	US	1 091 796.58	129 789.43	2	<0.01
		China	986 168.64			
		Russia	609 860.16			
	Tentat (= tentative)	US	901 724.90	45 713.87	2	<0.01
		China	1 274 567.20			
		Russia	1 014 470.81			
	Certain (= certainty)	US	1 185 959.35	23 738.47	2	<0.01
		China	911 483.31			
		Russia	1 024 929.72			
Work and Achievement	Work	US	1 052 951.62	18 752.69	2	<0.01
		China	868 639.51			
		Russia	1 039 959.41			
	Achieve	US	1 297 399.10	50 916.56	2	<0.01
		China	959 026.02			
		Russia	1 005 742.90			
Leisure, Social, and Emotions	Leisure	US	980 926.09	85 316.75	2	<0.01
		China	1 296 347.23			
		Russia	1 003 669.93			
	Social	US	465 536.27	226 454.01	2	<0.01
		China	1 110 495.82			
		Russia	985 566.56			
	Posemo (= positive emotion)	US	991 632.14	9 663.09	2	<0.01
		China	1 147 536.88			
		Russia	1 021 808.04			
	Negemo (= negative emotion)	US	1 059 339.80	30 543.89	2	<0.01
		China	1 053 508.36			
		Russia	849 192.81			

information exchanges, which was higher than their US (131) and Russian (24) counterparts. This finding was somewhat paradoxical, given that Chinese developers engaged in information exchange the most frequently, yet were the least inclined to provide answers to direct questions, as noted earlier in this section. Overall, information exchange ranked fourth among Chinese developer behaviours at 11.1%. Conversely, Russians were the least likely to engage in information exchange and this category ranked eleventh (1.43%) in their interaction and knowledge sharing behaviours. Information exchange ranked sixth (7.15%) among US developers, suggesting they regularly engaged in information exchange. The chi-square results for the Information Exchange category revealed statistically significant differences: $X^2 = 130.06$, $df = 14$, $p < 0.001$.

Chinese developers were also the most likely to engage in **discussion**. Chinese developers initiated a total of 531 discussions, and discussion ranked second (25.07%) in their interaction and knowledge sharing behaviours. The Chinese preference for initiating discussion was higher than that of US (333) and Russian (380) developers. Among Russian developers, the discussion category ranked second (18.17%), while among US developers, discussion ranked third (22.63%). Discussion was ranked in the top three across all three countries. The chi-square results for the Discussion category revealed statistically significant differences: $X^2 = 63.25$, $df = 26$, $p < 0.001$.

Comments ranked first among the interaction and knowledge sharing behaviours expressed by developers from all three countries. However, Chinese developers led this category by providing a total of 580 comments (27.38%). Russian and US developers provided a total of 510 (25.61%) and 370 comments (20.19%), respectively. The fact that comments ranked first across all three countries indicates a similarity among the developers of the three countries and the way they interacted on Stack Overflow. The chi-square results for the Comments category revealed statistically significant differences: $X^2 = 97.1$, $df = 24$, $p < 0.001$.

China led the **scaffolding** category as well. Scaffolding is the activity of providing advice to others. Chinese developers scaffolded a total of 310 times, and scaffolding ranked third (14.64%) in their interaction and knowledge sharing behaviours. Scaffolding also ranked third (9.71%) among Russian developers, who scaffolded a total of 163 times. Finally, US developers scaffolded a total of 73 times, and scaffolding ranked eighth (3.98%) among their interaction and knowledge sharing behaviours. The chi-square results for the Scaffolding category revealed statistically significant differences: $X^2 = 118.13$, $df = 14$, $p < 0.001$.

Chinese developers gave praise and expressed **gratitude** more often than developers from the other two countries. Gratitude ranked fifth (5.62%) in the interaction and knowledge sharing behaviours of Chinese developers. Chinese developers made a

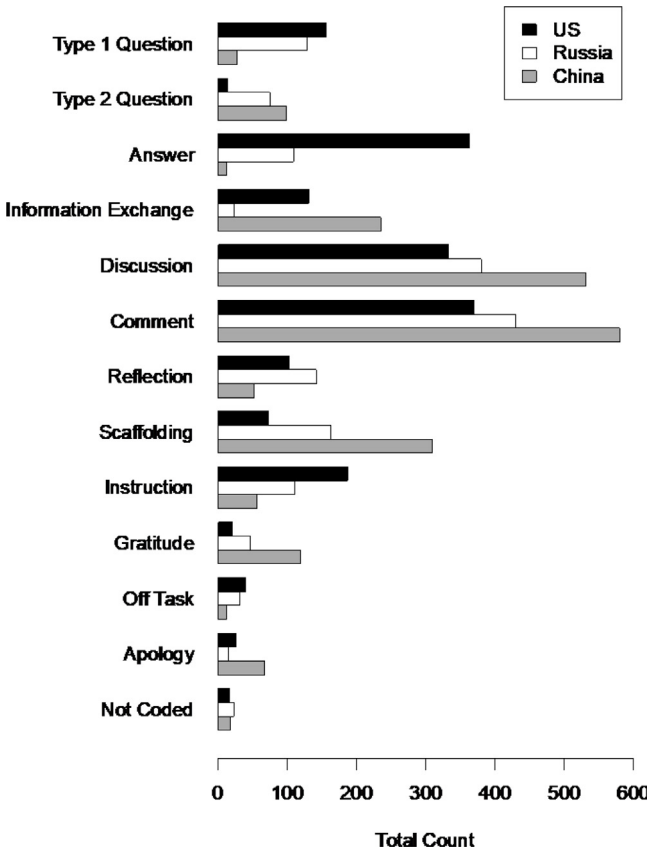

Fig. 2. Developers' interaction and knowledge sharing patterns across countries.

total of 119 expressions of gratitude, which was considerably higher than US (21) or Russian (47) developers. Gratitude ranked ninth (2.8%) among the 13 behavioural categories for Russian developers. US developers expressed gratitude the least frequently, and gratitude ranked eleventh (1.15%) among US developers. The chi-square results for the Gratitude category revealed statistically significant differences: $X^2 = 122.65$, $df = 8$, $p < 0.001$.

Chinese developers also led the **apology** category. Specifically, Chinese developers made a total of 67 apologetic responses and this category ranked seventh (3.16%) among their interaction and knowledge sharing activities. Conversely, US (26) and Russian (15) developers used this category infrequently. Apology ranked tenth (1.42%) and thirteenth (the lowest at 0.89%) among US and Russian developer behaviours, respectively. The chi-square results for the Apology category revealed statistically significant differences: $X^2 = 81.76$, $df = 6$, $p < 0.001$.

Russian developers were found to be more self-evaluative or **reflective** than developers from the other two countries. Russian developers made a total of 142 self-evaluations, and reflection ranked fourth (8.46%) among their 13 behavioural categories. US developers made a total of 102 self-evaluations, and reflection ranked seventh (5.56%) among their interaction and knowledge sharing behaviours. Despite Chinese developers' willingness to share knowledge, provide comments, and generate discussions, Chinese developers practiced self-evaluation less intensively than their US and Russian counterparts, providing only 52 reflections, and reflection ranked ninth (2.46%) in their interaction and knowledge sharing behaviours. The chi-square results for the Reflection category revealed statistically significant differences: $X^2 = 54.95$, $df = 14$, $p < 0.001$.

The categories used in this study adequately represented developers' interaction and knowledge sharing behaviours across the three different countries. The **not coded** category captured any behaviours that were beyond the scope of the other 12 interaction and knowledge sharing categories. The results indicate the frequency and percentage of behaviours assigned to this category was low across all three countries. For US developers, there were 17 not coded behaviours (0.93%), suggesting that the remaining 99.07% of behaviours were successfully categorised. Among Russian developers, the not coded category was also low and included a total of 27 behaviours (1.37%), which implies that 98.63% of behaviours were assigned to the established categories. Finally, the not coded category among Chinese developers consisted of 18 instances (0.85%), which implied that 99.15% of behaviours were able to be categorised. Overall, our chi-square results demonstrated that all categories of interaction and knowledge had statistically significant differences at <0.1% significance levels, indicating that developers across US, Russia, and China expressed interaction and knowledge sharing behaviours differently.

Given that the posts sampled were conceptualised as questions and answers on Stack Overflow, and the coding scheme also included these categories, we examined our outcomes to see whether these default labels may have confounded the pattern of results noted for Type 1 Questions, Type 2 Questions and Answers. We checked the codes returned for these categories against the distribution of records that were sampled (e.g., 180 questions and 204 answers for the US). We confirmed that the pattern of outcomes was not affected by the number of questions and answers that were sampled. For example, the proportion of Type 1 Questions asked by US developers against the total sample of question posts that were selected for our content analysis is 0.87, while for answers that number is 1.78. This contrasts with China, where the results are 0.15 and 0.06, and Russia where the results are 0.65 and 0.59 respectively.

In summary, the comparison of the frequencies (Fig. 2) and percentages (Fig. 3) demonstrated that US developers were the most likely to ask questions directly (Type 1 Question), provide Answers to knowledge seekers, provide Instruction to others, and express unrelated Off Task communications. Russian developers were the most likely to engage in self-evaluation (Reflection) during knowledge exchange. Finally, Chinese developers were the most likely to ask indirect questions (Type 2 Question), share knowledge with others (Information Exchange), engage in Discussion, provide Comments, express Gratitude for the help of other members, and offer Apologies to other members. Furthermore, Fig. 3 illustrates several noticeable trends among the three countries. More than half the interactions of Chinese developers (52.45%) were Comments and Discussions, which were the subject of slightly less than half (48.24%) of the interactions of Russian developers, and only 38.36% of the interactions of US developers. Also, 11.1% of the interactions of Chinese developers were to exchange knowledge with other members (Information Exchange), which constituted a considerably smaller share of the interactions of US and Russian developers, at 7.15% and 1.43%, respectively. Conversely, nearly one in five of the interactions (19.80%) of US developers were Answers to direct questions, which constituted only 6.49% and 0.57% of the interactions of Russian and Chinese developers, respectively.

5. Discussion and implications

We revisit our three research questions (RQ1, RQ2 and RQ3) in this section in discussing our findings and their implications.

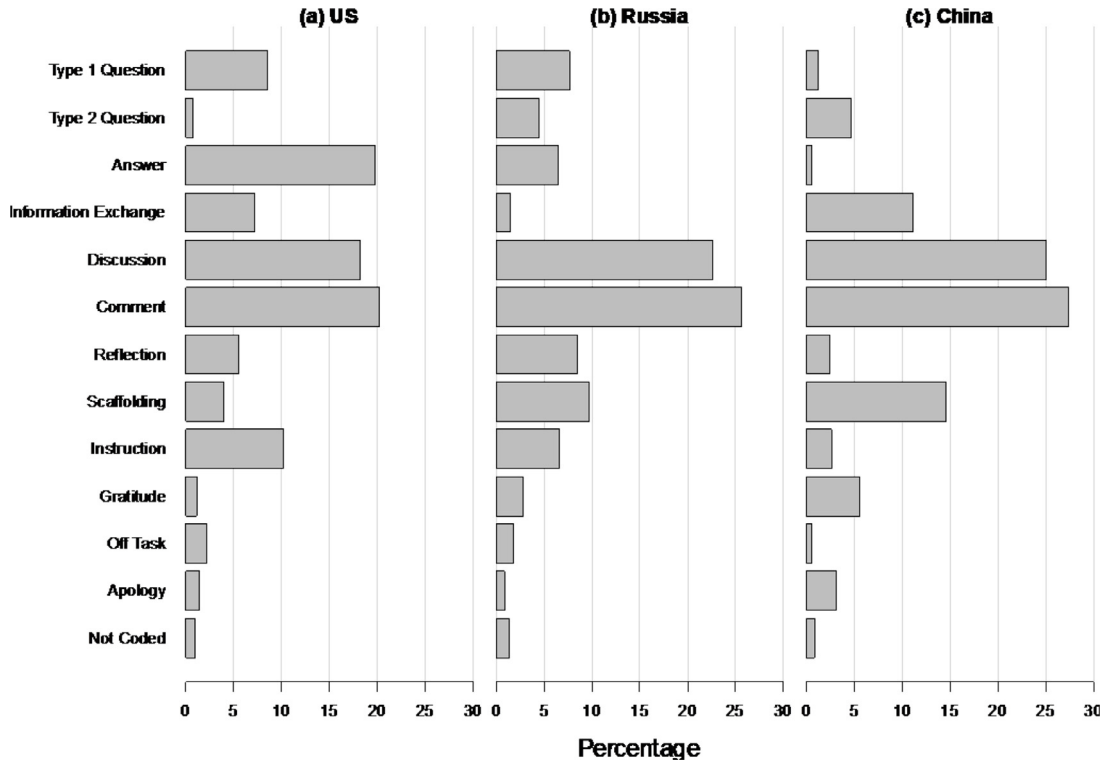

Fig. 3. Developers' interaction and knowledge sharing patterns within countries.

5.1. Developers' orientation (RQ1)

In this section, we analyse our findings with respect to RQ1: "What is the orientation of software development practitioners, on Stack Overflow, from collectivistic cultures compared to that of their counterparts from individualistic cultures?" This study explored the cultural orientation of software developers from the US, China, and Russia — three countries that differ considerably on Hofstede's cultural dimension of individualism. In summary, we found significant differences in the length of the self-description orientation among the three groups of developers, with US developers providing significantly longer self-descriptions than their Russian and Chinese counterparts. This finding suggests that US developers may be more expressive than their Russian and Chinese counterparts in describing themselves in a way that enhances their public image as technology leaders. As Hofstede et al. (2005) reported, countries ranking higher on the individualism dimension put more emphasis on personal accomplishments, which can be highlighted in self-descriptions. Conversely, Chinese and Russian developers may have lower interest in self-presentation since it is not a priority in their more collectivistic cultures. Also, the need to write a self-description in English may create a language barrier that serves to further discourage Chinese and Russian developers from creating a self-description (Wilczewski et al., 2017). To help their employees network more effectively, software companies from more collectivistic cultures may wish to show their developers how beneficial self-descriptions can be and encourage them to create these self-descriptions in order to benefit further from the Stack Overflow platform (Fox, 2019). If writing in English is a barrier, companies might want to provide translation services so that their employees can craft self-descriptions and use them to increase interest in their opinions or future collaborations.

The actual content from the developers' self-descriptions likely influences the number of views they receive. Our findings indicate that US developers received significantly more profile views than Russian and Chinese developers. The longer self-descriptions of US developers may translate into better marketing, which is consistent with the high number of profile views they received. As a result, US developers may be viewed by more potential clients and have a wider variety of job opportunities from which to choose. The use of self-descriptions allows software developers to leverage a broad audience for publicising information about innovative projects they and their companies are currently working on, allowing them to generate interest in future products. Self-descriptions also allow developers to create an extensive social network for collaborating with other developers and keeping abreast of developments in the field.

Although US developers had longer profiles and more profile views, Russian developers had been using the Stack Overflow platform for the longest period, with an average of slightly more than three years on the platform. This finding is reflected in the fact that Russia is one of the highest-ranked countries in the world based on the proportion of its people who use online technologies (Bank, 2019). Chinese developers had been using the platform for about 10–12 months less than the developers from Russia and the US, suggesting that Chinese developers may have less-developed social networks. A targeted and focused approach to developing the social networks of Chinese developers, including an emphasis on further developing their self-descriptions, may help them bridge the gap to the social networks of their Russian and US counterparts.

US developers had higher values than their Russian and Chinese counterparts in several orientation dimensions. Specifically, they asked and answered more questions and received more up votes than Russian and Chinese developers. US developers' individualism may cause them to stress the importance of individual tasks, making them nimble in asking and answering questions and up voting or receiving such votes. In the future, developers from more collectivistic cultures may become more quickly acclimated to the social features of Stack Overflow if they are provided with an initial explanation as to why the features are

used and how some developers have initially had more practice in their use simply because of their cultural backgrounds (Chen and Lin, 2020). This explanation may encourage developers to respond more actively to individual questions and focus less on conforming to the group decision-making process and social norms emphasised in collective cultures (Erez and Earley, 1993).

Research suggests that individuals from collectivistic cultures are not only more group oriented but also less trusting of other group members (Liu et al., 2019), which could manifest itself in online behaviour as avoidance of asking or answering questions in a way that allows other developers to judge them harshly or unfairly (such as through down voting), and thus avoid embarrassment. Consequently, developers from collectivistic cultures may see the benefit of interacting with online platforms as a small community of people, rather than as individual developers. Future research is needed to explore whether developers from collectivistic cultures are more likely to engage in asking and answering questions and voting as they gain trust in other group members and gain experience in using online platforms like Stack Overflow.

Consistent with the finding that US developers posted and responded to more questions and received more up votes than their Russian and Chinese counterparts, US developers also accumulated higher reputations and more badges. To the extent that software development firms wish to be superstars within their technical specialisation and become well-known within the entire development community, access to US developers is likely to be a huge asset. However, the ability to excel individually is not synonymous with having outstanding group work skills. If software development firms wish to identify team players who work exceptionally well with others, they may need to look beyond developers' online profiles and accomplishments to also consider their social reputation among developers they have worked with on previous projects. This is particularly necessary for software development efforts that depend on teamwork.

Although US developers posted the most questions and answers, they edited and commented on posts the least. Developers from China provided by far the most comments and edits. This finding may reflect the collectivistic nature of Chinese developers, in which they strive towards a well-integrated community. In commenting and editing, Chinese developers may have embellished the posts or added contextual information consistent with the norms of the high-context Chinese culture (Hall, 1976), in which communication tends to be indirect. For example, developers from collectivistic cultures may prefer to say, "perhaps it is better to consider the following", which is indirect and not self-focused. High-context cultures are also sensitive to disagreement that is indirectly expressed and seek to resolve it before moving forward, which may have compelled them to edit or comment to decrease potential conflict.

Conversely, communication is shorter, more direct, and highly task-oriented in the low-context US culture. Developers from individualistic cultures may be more demanding than those from collectivistic cultures, as reflected in their use of demanding words such as "should", "prefer", "needed", "problem", and "regardless". Examples of communication from individualistic cultures include "you shouldn't use regex for this purpose", "XML-style markup should never be processed with regular expressions", and "All great answers, however, a bit difficult for newbies. I assume you have learned the return statement".

Similar findings have emerged from prior studies, suggesting that members from collectivistic societies prefer indirect communication, which has a favourable impact on the decision-making processes (e.g., Gudykunst (1997), Oliveira et al. (2016) and Oliveira et al. (2018)). At the same time, people from individualistic cultures are more confrontational and direct in their communication (Erez and Earley, 1993). The next section addresses the attitudes of software developers in relation to individualistic and collectivistic cultures, developing a more nuanced understanding of the differences in communication styles revealed.

5.2. Developers' attitudes (RQ2)

Similar to Section 5.1, in this section we analyse our findings with respect to RQ2: "Do the attitudes of software development practitioners, on Stack Overflow, from collectivistic cultures differ from those who come from individualistic cultures? If so, what is the magnitude of these differences?" The second research question examined the attitudes of developers from individualistic versus collectivistic cultures. The results reviewed in this section are mostly exploratory, and the goal of the discussion is to promote an understanding of how developers from different cultures behave in social settings, not to pass judgement on their behaviours. Ultimately, the findings could be used to improve development environments and community structures for collaboration in software engineering, and should not be used to assess one culture as better than another.

Our first set of findings from the linguistic analysis concerned developers' pronoun usage. We found that developers from the US used the word "I" more frequently than developers from China and Russia. This outcome aligns with the findings from prior studies reporting that the comparatively high utilisation of self-references is related to individualistic demeanour (Licorish, 2013). It suggests that developers from the US, a country known for its individualistic culture, are more self-focused than developers from China and Russia, which have more collectivistic cultures. Our outcomes also suggest that developers who exhibit more individualistic attitudes depend heavily on themselves to maintain work tasks, meaning they are more task-focused. Developers who are task-focused tend to occupy a central position within the software engineering community by maintaining a neutral balance among all community members (Licorish, 2013). Our findings also revealed that Russian developers are less individualistic than US developers but more individualistic than Chinese developers, suggesting that Russian developers may be well-positioned to serve as mediators between developers with individualistic and collectivistic attitudes.

Chinese developers dominated in the use of the pronouns "we" and "you", which aligns with the findings from previous studies reporting that the use of first-person plural pronouns such as "we" among individuals with close relationships is most prevalent in collectivistic settings (Licorish and MacDonell, 2013a). Employing the second-person pronoun "you" in communications can imply the extent to which participants of a group depend on each other (Pennebaker et al., 2003). The results indicate Chinese developers are more reliant on one another than developers from the US and Russia. This inter-reliance among Chinese developers may manifest itself as more intense cooperation, reflecting a group focus rather than self-focus. Russian developers were between Chinese and the US developers in their frequency of using the pronouns "we" and "you", suggesting that Russia may have a balanced or mixed culture in terms of the individualism dimension. Overall, the pattern of findings revealed in the pronoun usage of developers is consistent with the individualism dimension of Hofstede's cultural dimensions theory (Hofstede, 1984).

Our second set of findings concerned developers' use of cognitive language. The results of André et al. (2011) indicated that the usage frequency of cognitive words is correlated with practical task analysis and brainstorming capabilities. Our analysis showed that Chinese developers were the most frequent users of

insightful words (e.g., think, believe, consider, determined, admitted, idea) that are typically used during group activities such as discussions, problem-solving sessions, or confirmation processes. The tendency of Chinese developers to use more insightful words indicates how they prioritise group coordination. This finding is consistent with Hofstede's cultural dimensions theory, which suggests that collectivistic cultures are more group-focused than individualistic cultures (Hofstede, 1984).

Developers from China also used tentative words (e.g., maybe, perhaps, apparently, chance, appears, hopeful) more frequently than developers from Russia and the US. This finding suggests that Chinese developers were more tentative in their actions and used tentative words more often to express ideas and suggestions indirectly in their software development artefacts. Conversely, US developers used certainty words (e.g., definitely, always, extremely, absolute, and certain) significantly more frequently than developers from China and Russia. US developers also used more discrepancy words (e.g., should, prefer, needed). The use of certainty and discrepancy words represents direct, unquestionable, and confrontational attitudes along with an individualistic demeanour. Examples of communication from individualistic cultures include: "I prefer you do the following" or "You should add bracket to your input statement", which are direct, demanding, and self-focused statements.

Our third set of findings was for the work and achievement linguistic category. Examples of work words include job, feedback, majors, goal, overtime, programme, and duty. Achievement-related language includes words like accomplish, win, attain, success, closure, overcome, resolve, better, obtain, solve, finalise, and fulfil. Evidence suggests that individuals who are interested in task completion use work and achievement words more often than those who do not share the same concerns (Benne and Sheats, 1948). Consistent with Hofstede's theory (Hofstede, 1984), developers from the US dominated the use of work and achievement words. For example, a US developer asked, "How do I obtain higher reputation through feedback to be in the top contributors release?"

Developers from collectivistic cultures may use words reflecting task completion less frequently merely due to their indirect communication style, which is highly contextual and less task-focused overall. Alternatively, the tendency of developers from collectivistic cultures towards a group focus may lead them to de-emphasise achievement and other topics that detract from or diminish the group's collective focus (Hofstede, 1984). Based on their linguistic patterns, US developers were found to be more concerned with achievement, perhaps driven by the desire to position themselves as leaders in their local and global communities. It may also be because they work in a competitive environment in which individuals are required to operate at their highest levels to sustain their positions within an organisation.

Previous work has noted that highly ambitious individuals tend to be results-driven (Denning, 2013) and are often dedicated to achieving success in any given situation (Chang et al., 2013). Well-established role theories and documented role principles have demonstrated how task-driven people aid in moving software development processes towards defined objectives and goals (Belbin, 1985). Thus, the results of previous studies are consistent with the current findings, which indicated that developers from cultures that ranked highly on Hofstede's individualism dimension were highly self-focused and outcome-oriented, as evidenced by their high usage of task-, work-, and achievement-related language.

Our fourth set of findings was related to the use of three linguistic categories: leisure, social, and positive emotion. The leisure category (e.g., club, movie, cinema, entertain, gym, golf, party, jog, film) is the opposite of the work category and is employed to emphasise non-task-related attitudes (Chang et al., 2013). Also, individuals who use social words (e.g., give, buddy, love, explain, friend, hey, inform, meet, pal) and positive emotion words (e.g., beautiful, relax, perfect, glad, eager, fantastic, luck, impress, proud) are said to exercise their group skills to enhance team atmosphere (Zhu, 1996). Our findings revealed that Chinese developers used leisure, social, and positive emotion words more frequently than Russian and US developers. For example, one Chinese developer commented, "It ranks in importance below many things such as family, my girlfriend, friends, happiness etc., and below other things I'd rather be doing if I had an unlimited supply of cash such as riding motorbikes, sailing yachts, or snowboarding". Chinese developers might also express positive sentiment, for instance, by saying, "write some inline sql (feels good), and improve requirements". Through this type of communication, developers from China may have demonstrated attitudes that strengthened group cohesion and generated more positive emotions among their colleagues. Research has shown that group moods supporting group optimism usually promote group satisfaction and cooperation, particularly in collective, social, and encouraging processes. Such motivational dynamics have a significant positive effect on the productivity and performance of the community (Bakas et al., 2020; Denning, 2013). A social climate may encourage members to contribute and might also build rapport (Chang et al., 2013). These findings suggest that developers from collectivistic societies may be particularly skilled at fostering positive moods among the group (Yao et al., 2020). For example, another Chinese developer noted, "Process always gets in the way of a good developer, no matter how Agile it is". Such statements could foster group cohesion and may ultimately lead to higher group productivity.

Finally, the fifth set of findings was for the linguistic category of negative emotions. Negative emotions are expressed through words such as afraid, hate, dislike, shock, stupid, and terrified. This type of linguistic pattern is commonly employed to scrutinise personnel that make a negative contribution to the social behavioural ambience of the community (Rastogi and Nagappan, 2016). Findings from this study demonstrate that US developers were the most frequent users of words indicative of negative emotions. The expression of negative sentiment may have an adverse impact on group coherence and community ambience. For example, a US developer responded to a post requesting expertise with, "BTW, I HATE eclipse", while another noted, "I can't believe no-one mentioned the 'goto case' syntax, I hate that one!" Such strong statements may be perceived as confrontational and biased for someone with alternative views (e.g., a contributor that likes Eclipse). The high use of negative emotion words by US developers may be related to the fact that the US is a highly individualistic country where citizens tend to be self-focused and less concerned with group climate and cohesiveness. Chinese developers had the second-highest frequency of negative emotion words. A potential disadvantage of living in a highly collectivistic society with strong cultural customs is a sense of deprived freedom that may be reflected through negative linguistics. It is said that those who express substantial negative emotions tend to be unsatisfied and distressed (Denning, 2013), which may explain why Chinese developers displayed more negative emotion words than Russian developers. This assessment is particularly plausible given that Chinese contributors were also the most social and positive. In general, a substantial body of scholarly research has shown that negative moods have a deleterious effect on group work outcomes (Chebat et al., 2010). Software developers should be aware that regardless of the cultural background of the group, a positive mood among community members tends to make them not only more cohesive but also more productive, while negative moods have the opposite effect.

5.3. Developers' interaction and knowledge sharing patterns (RQ3)

Finally, in this section we discuss our findings for RQ3: "How does the process of interaction and knowledge sharing compare between software development practitioners, on Stack Overflow, from collectivistic cultures versus those from individualistic cultures?" Our third research question explored the interaction and knowledge sharing patterns of developers. Through content analysis, our findings revealed that developers from individualistic cultures (e.g., the US) were more likely to ask direct questions to resolve a knowledge deficit and provide answers in response to such direct questioning. Developers from collectivistic cultures (e.g., China) were more likely to ask and respond to indirect questions. Developers from collectivistic cultures were also more likely to engage in knowledge sharing and discussion. These findings might indicate that developers from collectivistic cultures are more likely to engage in group-building and take steps to increase the cohesion of software engineering communities. When examined together with the findings for developers' orientation (RQ1) and attitudes (RQ2), these outcomes are compelling and point to the relevance of human behaviours.

Our results indicate that US developers were significantly more likely to ask type 1 questions and provide answers than their Russian and Chinese counterparts. This may be because their individualistic culture values direct and task-focused behaviour. Conversely, Chinese developers asked more type 2 questions and posted more comments, perhaps because their collectivistic culture values indirect questioning and has a group focus. For example, a US developer asked the following direct question: "Which class contains the implementation and deployment for screen hibernation feature?" Conversely, a Chinese developer instead initiated the discussion of a question, posting "let us talk about the new Python method that discards the unique index and has even weirder side effects in more complicated cases". Our outcomes align with those of previous research suggesting that developers from individualistic cultures favour direct and confrontational communications, while developers from collectivistic cultures prefer indirect communications (Hofstede et al., 2005). These findings may be particularly relevant for building software engineering groups and communities, as well as encouraging the development of relationships among software engineering professionals.

Next, we explored cultural differences within the following categories of knowledge sharing behaviour: information exchange, discussion, and reflection. Information exchange occurs when developers announce or share knowledge, and discussion is the process of elaborating or expressing ideas. Within these two categories, developers from China appeared more frequently than their US and Russian counterparts, indicating that members from a collectivistic culture were more likely to share information with other developers and discuss implications of the information shared, which is consistent with the findings of previous studies (Arpaci and Baloğlu, 2016). Different findings emerged for the category of reflection, in which developers engage in self-evaluation. Developers from Russia engaged in reflection more frequently than those from the US and China. Previous research suggests that self-reflection or "brooding" is a characteristic commonly associated with Russian culture (Grossmann and Kross, 2010). Russians were more likely than Americans to demonstrate adaptive, flexible self-reflection in which they self-distanced while analysing their feelings about an event, thereby helping them avoid distress, anxiety, or depression as a result of their reflection.

Overall, these findings indicate that developers from collectivistic cultures are more active in exchanging or transferring information. Their behaviours can be explained by a group mentality typically inherent in cultures with lower ranks of individualism (Hofstede, 2016). These findings also align with prior evidence indicating that developers from China were more engaged in leading discussions than developers from the US, particularly discussions in which information, ideas, and thoughts were exchanged as part of the interaction and knowledge sharing processes. For example, a user from China posted "Hi I am studying laravel. I use Eloquent ORM delete method but I get a different result. Not true or false but null. I set a resource route and there is a destroy method in UsersController". The user then leads the exchange further with contextual information: "thanks for the note. I use the query builder (where) and it works. Then I try User::destroy($id) it returns 0. But the record in the table disappear. I have a foreign key on id, is that the reason that I get a zero result while the deletion works?" Here quite a bit of information is shared in contextualising the question. Chinese developers also demonstrated the highest frequencies for the categories of apology (e.g., expressing remorse) and gratitude (e.g., giving appreciation). These findings may be related to the fact that China is a collectivistic culture that is group-focused and tends to assign higher priority to group collaboration than individualism (Hofstede et al., 2005). Showing appreciation and giving apologies are essential to maintain group cohesiveness.

We also explored cultural differences in the instruction and scaffolding categories. Instruction refers to giving directions or commands, while scaffolding refers to the process of providing a proposition or advice. While these categories are similar, the findings differed by culture. US developers were the most likely to give commands or instructions. For example, a US developer offered the following instruction "Use typedef to define your data type before running the main script". Conversely, Chinese developers were the most likely to provide scaffolding. Russian developers ranked in the middle in both categories. These findings are consistent with Hofstede's cultural dimensions theory (Hofstede et al., 2005). The dominance of the US contributors in providing instruction that is more authoritative, less friendly, and more individualistic is consistent with the high ranking of US culture on the individualism dimension. By contrast, giving advice is less authoritative and friendlier in a more collectivistic culture, thus maintaining community collaboration. Overall, these results converged with the cognitive and social linguistic categories revealed throughout the linguistic review stage of this research.

Finally, we explored the off task category, which comprises communication unrelated to the task or post at hand. Our findings indicate that US developers engaged in more off task communications during interaction and knowledge sharing than developers from China and Russia. This finding may indicate that self-focused individuals are more susceptible to stating what is on their minds. For example, a US developer noted, "I love programming in assembly language, but it takes more code to do the same thing as in a high-level language, and there is a direct correlation between lines of code and bugs (this was explained decades ago in The Mythical Man-Month)".

In terms of communications not captured by our protocol, developers from Russia had the highest scores for the not coded category (scoring 23, whereas China scored 18, and the US scored 17). This finding indicates there could be a few other interaction and knowledge sharing categories beyond those previously defined.

5.4. Implications for software developers and software development organisations

While Stack Overflow is a Q&A platform and does not necessarily involve collaboration in the sense of software development

projects, many of the findings in this work are relevant for software development team settings. On its website, Stack Overflow describes itself as a place where "everyone who codes" comes "to learn, share their knowledge, collaborate, and build their careers".[6] In this sense, the community engagement that takes place on Stack Overflow is very much a form of collaborative engagement. Thus, this study has a number of implications for software developers and software development organisations. Understanding the role of culture and significant cultural differences can help to overcome barriers to collaboration between software developers and improve products by utilising the skill-sets of team members from diverse cultural backgrounds (Kunst et al., 2021). The findings from this study reveal a number of key approaches that can harness the power of cultural difference in collaboration rather than attempt to reduce these differences and, thus, lose potential elements that can greatly enhance a project's results.

First, in terms of software developers' orientation, developers from more individualistic cultures tended to ask and answer more questions, while developers from more collectivistic cultures tended to reflect more on their posts and carefully edit them. For teams composed of members from diverse backgrounds, it would be helpful to recognise these different approaches to the way solutions are provided. The direct asking and answering of questions by individualistic team members might be a great place to start and could be complemented by the collectivistic team members, which are likely to be helpful in honing a team approach to problem solving. Furthermore, recognising the need for team members from collectivistic cultures to build their trust in the team would benefit team members from all backgrounds and build a stronger, more cohesive team that can effectively address the software development challenges it faces.

Second, in relation to software developer attitudes, developers from individualistic cultures tended to be more results-driven while those from collectivistic cultures tended to be more focused on the cohesiveness of the group and how to work together. With team members from both backgrounds, a balance can be struck between achieving results and collaborating to create the best possible product. Furthermore, developers from collectivistic cultures tended to express more positive emotions, aiming to improve group cohesiveness, which means that including team members from collectivistic cultures might improve collaboration and team morale, which have been found to lead to better results (Denning, 2013).

Third, this study found differences in interaction and knowledge sharing patterns. Developers from collectivistic cultures tended to participate more in knowledge sharing, which would greatly benefit software development teams working to find the best possible solution. Developers from individualistic cultures tended to ask more direct and task-focused questions, which is beneficial for keeping the team focused and working towards results. A team is likely to benefit from the diversity of cultural background, as this diversity would allow for collaboration that is both results-oriented and reflective.

Beyond harnessing the positive aspects of individualistic and collectivistic cultures, software development organisations could also use these findings to strategize towards improving their members' approaches to collaborative software engineering. For instance, organisations might implement cultural training seminars that give developers skills to improve their interactions within a diverse team. Organisations might also consider team-building activities to build trust and support members from collectivistic cultures. Team performance can be enhanced through the awareness of cultural differences and support can be provided as necessary to develop the best possible product.

[6] https://stackoverflow.com/company.

6. Study limitations

There are several potential limitations to this study. First, the developers' artefacts were obtained from Stack Overflow, which may not represent all developers' communications, some of which may be exchanged via private messages between community members. Additionally, because there is a Russian-language Stack Overflow platform (Stack Overflow на русском), Russian developers may be choosing to post primarily in their native language (Bachschi et al., 2020). This means that many of the Russian developer posts might skew towards the Russian platform and thus not be included in this data. It may also mean that the Russian developers participating on the English-language Stack Overflow might be a specific subset of Russian developers, such as specialist developers or developers on international teams. Thus, the data used in this study may not fully reflect the orientations, attitudes, or interaction and knowledge sharing patterns of all Russian developers. Second, developers' orientation dimensions were measured using a frequency-based approach obtained from their profile and contributions on Stack Overflow. Additionally, developers' orientations may not be uniform, as some developers might contribute to questions considered more technical than others. Future research should consider contribution complexity when investigating developers' orientations. Third, the archival repository data examined in this study was not primarily prepared for research purposes, which might act as a threat to the internal reliability of this study (Runeson and Höst, 2009).

In terms of other threats, there are also a number of limitations related to our use of the GeoText tool. We discovered that it assigns more weight to the first location reference like cities over other references (e.g., provinces) in detecting country rather than cross-checking both simultaneously to produce a country. For example, if a user included their location as 'Cairo, Illinois' then the country returned will be 'Egypt' rather than the United States, and such records were omitted from our final dataset. Also, if a user includes a city name that is not unique, then the tool will assign that particular city to the more prominent location rather than giving it a null value. Taking the same example, if the inserted location string contains only 'Cairo' then the tool will assign it to the country 'Egypt', despite the existence of a city with the same name in Illinois, United States. Additionally, the tool will assign a country to the first city mentioned in the inserted string rather than produce separate results. As demonstration, the location reference 'Austin, Paris' will be assigned to the United States rather than being null or assigning the entry to all possible countries ('Austin, United States' and 'Paris, France'). Furthermore, the tool struggles with identifying locations written in a language other than English, such as 'Москва' (Moscow) and '北京市' (Beijing). Finally, the tool seems to fail in identifying the country where a state or province code is found without a clear indication to a city name (e.g., '143 Ketcham St, AL'). That said, our reliability checks showed that the GeoText tool was accurate 82.03% of the times, albeit it was only used to identify 10,669 (7.6%) of the 140,184 contributors studied. Given that the remaining 92.4% of the contributors in our sample clearly identified their country of origin, we are confident that the use of GeoText does not represent a significant threat to our findings.

Regarding external validity, the sample may not be representative the entire population of developers within each country. Moreover, the development processes within a single country might be focused on a specific technology on the Stack Overflow platform, which may not be the focus on other development platforms (e.g., SourceForge) within the same country. The developers in this study utilised the Stack Overflow platform to communicate about software development subjects such as software coding

and design. This type of communication may be less mature when matched against communication from other software community groups such as Reddit, CodeProject, CodeRanch, Programmers Heaven, and FindNerd. Therefore, the current results may not be representative of all software development situations.

We performed a thorough manual check across 384 randomly-sampled records from our original dataset (over 10M) to manually assess if users/contributors add incorrect country of origin to their Stack Overflow profile. Of this sample, 210 contributors identified their location, while the others did not identify a location. Of the 210 contributors, 138 provided detailed information about who they are in their profile, including links that were verified (e.g., their CV/website). The remaining records were taken as given. That said, we have not studied records with missing country details, and given the outcomes of our analysis, we believe that a large amount of contributors in our sample were transparent about their country of origin. However, we cannot be sure that all users have identified their real location or nationality since some users might add incorrect details for unknown reasons.

Our results suggest a connection between the orientation scores of developers and their individualistic versus collectivistic cultural backgrounds. However, US developers are the only group to employ their native language while utilising the Stack Overflow platform. In contrast, Russian and Chinese developers are likely to be non-native English speakers who may construct sentences in ways that resemble the common forms of their native languages. Consequently, the native language of the developers can be a threat to validity.

Countries may also have exhibited differences on dimensions other than the individualism dimension that are not captured in this study. Similarly, another potential limitation of the study is the conflation of country with culture and the appropriateness (or lack thereof) of using Hofstede's cultural framework, which was developed based on studies of business managers as opposed to software developers. The present study recognises that any given individual may or may not reflect the overall aggregate culture of their country. At the individual level, there are multiple determinants of or influences on culture, such as the different groups to which a person belongs. These groups may be ethnic, familial, structural, organisational, and so on Beugelsdijk et al. (2017) and Hofstede (2016). A main goal of the study, however, was to examine large amounts of data to identify the broader patterns that emerge despite individual variation, based on the assumption that such trends in the data may supplant individual differences. A similar limitation exists in terms of the developers sampled and whether they live in urban or more rural areas. This also applies to the differences in living conditions between countries. For instance, the Chinese developers sampled may live predominantly in urban areas, while the US developers might be more divided between rural and urban. This could influence the results as orientation, attitudes, and interaction and knowledge sharing patterns may vary based on the type of areas in which the developers live. There is also the possibility that we had oversampled a particular geographic location within a country. This does not invalidate the analysis, but does provide salient avenues for future research.

While Hofstede's framework was not developed by studying software developers, it has a long history of use in cultural studies (Kirkman et al., 2006; Zhou and Kwon, 2020) and has also been the framework of choice in similar previous studies such as Gaspay et al. (2009). Future studies should consider alternative frameworks or models for the study of cultural influence on the behaviour of software developers in order to more fully assess the appropriateness of Hofstede's framework for this type of research. McSweeney (2002) critiqued Hofstede's framework, arguing that culture is not involved in determining behaviour. McSweeney (2002) bases his critique on the narrowness of the population surveyed by Hofstede. Because Hofstede based his findings on a survey conducted among individuals working for a single company, how could they be valid in a broader context? Furthermore, McSweeney (2002) argues that Hofstede's assumption of a uniform national culture is false. However, Hofstede (1984) did not presume cultural uniformity and his results ultimately showed much variance in culture with his model reflecting the average responses within a particular culture. Similar limitations apply to the current study, especially regarding the generalisations to different cultures. Future work should further explore the nuances within specific cultures to determine whether a uniform approach to managing people from different cultural backgrounds can improve software development.

It is also unclear as to whether the differences observed between groups in real life are significant enough to warrant specifically prescribed action based on the culture of the developers. The inferences from the findings of this study raise several concrete suggestions, but these findings may not be enough to develop management guidelines for different cultures. This is an excellent avenue for future research, especially case studies of international software development teams. These case studies could examine the orientation, attitudes, and interaction and knowledge sharing patterns of team members from different countries to help determine the effect of culture on the success and productivity of diverse teams.

Another potential limitation in this study is the scale of the effect sizes where statistical differences were observed. Multiple tests used in this study include mechanisms to account for effect sizes. When employing the Levene test, for example, using the same unit of observation ensured comparability across samples because the effect size is equivalent to the unit of measure (Baguley, 2009). When employing the Kruskal–Wallis test, the mean rank comparisons between countries are essentially the effect size comparisons (Mangiafico, 2016). However, future research should endeavour to be more precise in the handling of effect sizes by employing specific methods such as standardised mean differences, odds ratios, and correlation coefficients (Salkind, 2006).

Determining whether the three samples explored in this study are sufficiently independent of each other is another potential limitation. The Levene test was employed as a complementary test of the independence of the samples in addition to the Kruskal–Wallis test and the pairwise Mann–Whitney test. Checks were also done using the Brown–Forsythe test, which is a more robust variation of the Levene test (Mangiafico, 2016), producing similar results. The p-values for the 11 orientation dimensions in both tests indicated a less than 1% significance level. The advantages of employing the Brown–Forsythe test are that it does not assume a normal distribution of the populations studied and that it is well-known for its sensitivity to unequal sample sizes, making it especially well-suited to test the non-homogeneity and independence of the samples from the three countries. Although there was little difference in outcomes for our analyses between the Levene and Brown–Forsythe tests, future studies may investigate these tests for their precision when comparing cultural data from different samples. Furthermore, triangulating the findings from this stage using a more in-depth directed content analysis technique significantly reduced the risks to validity and reliability (Runeson and Höst, 2009). The content analysis for this study required the authors to analyse textual information that might be considered subjective, but multiple procedures were used to minimise these concerns. First, the coding protocol was adopted from previous work that employed and tested it for reliability and validity (Licorish, 2013). Second, the precision of this coding protocol was tested through a pilot session in which excellent

inter-rater agreement was obtained (Selvi, 2019). Runeson and Höst (2009) indicated that these procedures normally address reliability concerns of a case study. These efforts were undertaken to address methodological issues and strengthen the robustness of the findings.

Similar to the work of Espinosa et al. (2007) and Jaanu et al. (2012), this study provided evidence that cultural and geographical variations influence the behaviour of software developers operating on Stack Overflow. However, there might be subcultural differences within a culture that have a direct effect on software development processes, thus impacting developers' behaviour (Oyserman, 2017). Research examining the impacts of cultural diversity on software engineering has discovered several behavioural discrepancies between software developers operating within the same culture (Horta et al., 2019). Since the developers from the three countries in this study are naturally independent, this issue might have had a limited impact on the observed behavioural patterns. Potts (1993) indicated that substantive issues discussed on mass platforms are expected to be similar across contexts. However, case study findings cannot necessarily be generalisable to the entire population (Wikfeldt, 2016). Therefore, this research encourages future work to ensure that the findings from this work can be applied to a comparably large-scale population of software engineering processes. The results support a unidimensional conceptualisation of individualism and collectivism wherein profile patterns reflect a unidimensional framework. However, more research is needed to explore whether cultural heterogeneity is as significant within cultures as it is across cultures.

One final limitation that should be discussed is the application of the findings to (global) software development teams. As discussed in Section 5.4, the Stack Overflow setting overlaps with but is not the same as the setting that might occur on a team or within a given software development project. Though the community engagement is certainly collaborative and involves a group of diverse people working together to answer a specific question or solve a particular problem, there are differences between this online community and a software development team. Many of the findings are helpful for enhancing interactions with a group or team where collaborative engagements occur. However, future research should consider the extent to which these findings can apply in software development team settings.

7. Summary, conclusion and future research

Evidence from this study reveals that software developers from individualistic cultures favour direct communication to give instructions, as well as to produce and consume solutions, as it is found that developers from the United States had longer profile self-descriptions, posed more questions, provided more answers, provided more up and down votes, had higher reputations, and acquired more badges than developers from more collectivistic cultures like Russia and China. Users scoring highly on Hofstede's individualism dimension, were considered to be task- and achievement-focused and also favour direct communication through instructions and asking and responding to questions. These traits may enable more individualistic developers to serve as leaders who could govern software development communities and move technology forward. Linear to this argument, the individualistic mentality of American developers, who are self- and task-focused, also featured prominently in the orientation dimensions. In contrast, developers who score lower on Hofstede's individualism dimension, such as Russia and China, are collectivistic members who favour indirect means of communication such as comments and discussion to present ideas, and who can positively engage other community members through discussions and recommendations, making them effective in problem-solving. Results show that these developers were more concerned than the ones from the United States about maintaining group cohesion and norms. Consequently, developers from Russia and China spent a greater proportion of their time observing other community members' decisions and indirectly engaging when appropriate by commenting on and changing posts. Given the patterns presented by this research, project managers and community moderators in the development field should consider cultural factors carefully when forming teams so that a balance can be struck which benefits both the team and the project. They should also not be concerned if they note that a particular cultural group is dominating an aspect of development. Instead, they should recognise how each individual adds value to the development process in one form or another, as each member plays a significant role in the cohesion of the group. Project managers and community moderators should also encourage group members to be flexible and to engage in ways outside their natural proclivities. For example, members from an individualistic culture who happen to serve as leaders could be encouraged to engage with others through discussions and problem-solving activities, while members from a collectivistic culture could be provided with additional support from managers and community moderators when taking on leadership roles. This approach acts as a fail-safe mechanism by reducing the danger inflicted by the loss of a group member and their project knowledge, which can have a devastating impact on overall group performance. Furthermore, managers and community moderators should be watchful for situations where collectivist developers are reluctant to take leadership duties, and should provide appropriate mentorship and encouragement.

The offering of praise may be a helpful gesture from those in leadership roles to those in regular roles in order to boost their esteem in the group and indicate that higher management is recognising how their participation will ultimately enhance the performance of their entire development community. Also, since collectivistic developers tend to be positive communicators, it might be beneficial to include more achievement-driven individualistic team members to provide balance. In this way, the group might be steered towards a more results-focused approach while still maintaining a positive, collaborative environment. This trait of being achievement-driven, if developed, can eventually propagate to the entire development environment and may improve group performance. Negative attitudes in a software development environment, however, need to be discouraged. While negative attitudes are sometimes expressed by contributors due to frustration at being unable to achieve deadlines or meet standards. However, expressing them may propagate to other members and exacerbate the group climate, which, in turn, may negatively affect overall community performance. Supporting social and positive attitudes is an effective approach to enhance group and community performance. The findings from this study that developers from some cultures are highly task-focused while others are positive, social, and discussion-focused might not sound like an ideal group atmosphere. However, both types of individuals are equally essential, and should be encouraged in software development communities, as noted by Tjosvold et al. (2003).

In terms of interaction and knowledge sharing behaviour patterns, collectivistic developers display higher insight levels during their communications, which might profit junior community members or members coming from individualistic cultures. Findings also suggest that developers from collectivist cultures like China and Russia expressed more delegation and collectivist language while scoring highly in the social, leisure, and positive linguistic categories, implying that they rely on one another to foster

a social and positive community atmosphere. Thus, it might be a reasonable decision for project administrators to develop policies directed at promoting the involvement of less engaged developers by pairing them up with developers from a collectivistic culture. This strategy may help inexperienced community members to attain knowledge and mentorship. In contrast, developers from individualistic cultures such as the United States tend to be task-focused and may be more likely to display negative emotions in their communications. The expression of negative emotions may be due to overall disappointment with faulty communication between community members, or because members were unable to describe their problem adequately or provide a satisfactory solution. Thus, the cause of negative emotions should provide recommendations for precise communications between members, especially when the diversity of the group or community is high. In this study, we found that developers from collectivistic cultures present higher ratings in information exchange throughout their communication. Consequently, strategies need to be established to encourage communication between developers from individualistic cultures and skilled collectivistic communicators so that working knowledge is maintained when one or the other is unavailable. Project administrators should also pay attention to individuals who are too task-focused and encourage them to improve their communication skills, especially those who provide instructions regarding software tasks since these individuals often serve as central figures in software development (André et al., 2011). Project administrators should also be wary of the possibility of excessive dependence on these members by the rest of the community, which might exhaust and affect them in a negative way.

We believe the presence of collectivistic developers is crucial in the software development environment to reduce the risks associated with knowledge impairment if members leave. Collectivistic contributors also maintain a positive atmosphere in the development environment. However, there exists doubt about the "drive" of these developers (Graziotin et al., 2018). Although past work has investigated this matter (Curseu and Fodor, 2016), there is still a need for future face-to-face research to precisely examine collectivistic developers' incentives. Future research should combine interviews with repository data to provide further insights into the causes behind such patterns.

Finally, the findings in this research reveal that developers from individualistic cultures are highly self-focused and use more work- and achievement-related language. These outcomes provide evidence for the connection between an individual's orientation and their attitude, as discussed in social incentive theories suggesting how the increase in a particular behavioural pattern encourages another performance pattern (Graziotin et al., 2018). Additionally, developers from China and Russia were also more perceptive and cautious in their discussions than developers from the United States, who were more confident and direct. Still, the outcomes of this research were not absolute, so this question requires further research.

This study overcame the shortcomings of relying solely on one methodological approach by using both quantitative and qualitative methods. The employment of multiple robust analysis methods reveals how examining software developers' behaviours through their language use improves our understanding of software developers and their communities.

CRediT authorship contribution statement

Elijah Zolduoarrati: Conceptualization, Data curation, Methodology, Writing – original draft. **Sherlock A. Licorish:** Conceptualization, Data curation, Supervision, Writing – review & editing, Statistics validation. **Nigel Stanger:** Data curation, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was supported by an University of Otago Postgraduate Publishing Bursary, New Zealand.

References

Aaltio, I., Heilmann, P., 2010. Case study as a methodological approach: From locality to understanding the essence. Encyclopedia Case Study Res. 66–76.
Abreu, R., Premraj, R., 2009. How developer communication frequency relates to bug introducing changes. Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops, pp. 153–158.
Achinivu, G.K., 2017. The Influence of Culture on International Management: A Study of MAERSK Line Nig. Ltd. Covenant University, Ota, Nigeria.
Ali, M., 2020. Requirement gathering techniques widely used in global software engineering: A comprehensive study. Lahore Garrison Univ. Res. J. Comput. Sci. Inf. Technol. 4 (1), 1–9.
Ali, A.J., Taqi, A.A., Krishnan, K., 1997. Individualism, collectivism, and decision styles of managers in Kuwait. J. Soc. Psychol. 137 (5), 629–637.
André, M., Baldoquín, M.G., Acuña, S.T., 2011. Formal model for assigning human resources to teams in software projects. Inf. Softw. Technol. 53 (3), 259–275.
Ardichvili, A., Maurer, M., Li, W., Wentling, T., Stuedemann, R., 2006. Cultural influences on knowledge sharing through online communities of practice. J. Knowl. Manage..
Arpaci, I., Baloğlu, M., 2016. The impact of cultural collectivism on knowledge sharing among information technology majoring undergraduates. Comput. Hum. Behav. 56, 65–71.
Bachschi, T., Hannak, A., Lemmerich, F., Wachs, J., 2020. From asking to answering: Getting more involved on stack overflow. arXiv preprint arXiv:.04025.
Baguley, T., 2009. Standardized or simple effect size: What should be reported? Br. J. Psychol. 100 (3), 603–617.
Bakas, D., Kostis, P., Petrakis, P., 2020. Culture and labour productivity: An empirical investigation. Econ. Model. 85, 233–243.
Bank, T.W., 2019. Internet usage per country.
Bazelli, B., Hindle, A., Stroulia, E., 2013. On the personality traits of stackoverflow users. 2013 IEEE international conference on software maintenance, IEEE, pp. 460–463. http://dx.doi.org/10.1109/ICSM.2013.72.
Belbin, R.M., 1985. Management teams, why they succeed or fail. Bull. British Psychol. Soc. 38, http://dx.doi.org/10.4324/9780080963594, A1–A1.
Benne, K.D., Sheats, P., 1948. Functional roles of group members. Shared Exp. Human Commun. 155.
Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R., Zimmermann, T., 2008. What makes a good bug report? In: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 308–318. http://dx.doi.org/10.1145/1453101.1453146.
Beugelsdijk, S., Kostova, T., Roth, K., 2017. The impact of Kirkman, Lowe and Gibson's article on country level research in international business. J. Int. Bus. Stud. 48 (1), 30–47.
Bird, C., Murphy, B., Nagappan, N., Zimmermann, T., 2011. Empirical software engineering at microsoft research. In: Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work, pp. 143–150. http://dx.doi.org/10.1145/1958824.1958846.
Brown, M.B., Forsythe, A.B., 1974. Robust tests for the equality of variances. J. Amer. Statist. Assoc. 69 (346), 364–367.
Calefato, F., Lanubile, F., Marasciulo, M.C., Novielli, N., 2015. Mining successful answers in stack overflow. In: 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, pp. 430–433. http://dx.doi.org/10.1109/MSR.2015.56.
Casado-Lumbreras, C., Colomo-Palacios, R., Soto-Acosta, P., Misra, S., 2011. Culture dimensions in software development industry: The effects of mentoring.
Casey, V., 2010. Developing trust in virtual software development teams. J. Theor. Appl. Electron. Commer. Res. 5 (2), 41–58.
Chang, K.-c., Yen, H.-W., Chiang, C.-C., Parolia, N., 2013. Knowledge contribution in information system development teams: An empirical research from a social cognitive perspective. Int. J. Project Manage. 31 (2), 252–263.
Chebat, J.-C., Kerzazi, L., Zourrig, H., 2010. Impact of culture on dissatisfied customers: An empirical study. City Cult. Soc. 1 (1), 37–44.
Chen, S., Lin, N., 2020. Culture, productivity and competitiveness: disentangling the concepts. Cross Cult. Strateg. Manage..
Choi, K.S., Im, I., Hofstede, G.J., 2016. A cross-cultural comparative analysis of small group collaboration using mobile twitter. Comput. Hum. Behav. 65, 308–318.

Cohen, J., 1960. A coefficient of agreement for nominal scales. Educ. Psychol. Meas. 20 (1), 37–46.
Coltheart, M., 1981. The MRC psycholinguistic database. Q. J. Exp. Psychol. Sect. A 33 (4), 497–505.
Curseu, P.L., Fodor, O.C., 2016. Humor and group atmosphere: Development of a short scale for evaluating affiliative and aggressive humor in groups. Team Perform. Manage..
Dam, H.K., Tran, T., Pham, T.T.M., Ng, S.W., Grundy, J., Ghose, A., 2018. Automatic feature learning for predicting vulnerable software components. IEEE Trans. Softw. Eng..
Damerau, F.J.J.I.P., 1993. Management generating and evaluating domain-oriented multi-word terms from texts. 29 (4) 433-447.
Denning, P.J., 2013. Moods, wicked problems, and learning. Commun. ACM 56 (3), 30–32.
Deshpande, S., Richardson, I., Casey, V., Beecham, S., 2010. Culture in global software development-a weakness or strength? In: 2010 5th IEEE International Conference on Global Software Engineering, IEEE, pp. 67–76. http://dx.doi.org/10.1109/ICGSE.2010.16.
Elleson, V.J., 1983. Competition: A cultural imperative? Personnel Guid. J. 62 (4), 195–198.
Erez, M., Earley, P.C., 1993. Culture, Self-Identity, and Work. Oxford University Press on Demand.
Espinosa, J.A., Slaughter, S.A., Kraut, R.E., Herbsleb, J.D., 2007. Familiarity, complexity, and team performance in geographically distributed software development. Organ. Sci. 18 (4), 613–630.
Faul, F., Erdfelder, E., Lang, A.-G., Buchner, A., 2007. G* power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences. Behav. Res. Methods 39 (2), 175–191.
Fischer, R., Vauclair, C.-M., Fontaine, J.R., Schwartz, S.H., 2010. Are individual-level and country-level value structures different? Testing hofstede's legacy with the schwartz value survey. J. Cross-Cult. Psychol. 41 (2), 135–151.
Fox, S., 2019. Mass imagineering, mass customization and mass production: Complementary cultures for creativity, choice and convenience. J. Consum. Cult. 19 (1), 67–81.
Gaspay, A., Dardan, S., Legorreta, L., 2009. Software of the mind-a review of applications of hofstede's theory to IT research. J. Inf. Technol. Theory Appl. 9 (3), 3.
Gill, A.J., Oberlander, J., 2003. Perception of e-mail personality at zero-acquaintance: Extraversion takes care of itself; neuroticism is a worry. In: Proceedings of the Annual Meeting of the Cognitive Science Society, 25.
Glisson, C., James, L.R., 2002. The cross-level effects of culture and climate in human service teams. Int. J. Ind. Occup. Organ. Psychol. Behav. 23 (6), 767–794.
Graziotin, D., Fagerholm, F., Wang, X., Abrahamsson, P., 2018. What happens when software developers are (un) happy. J. Syst. Softw. 140, 32–47.
Graziotin, D., Lenberg, P., Feldt, R., Wagner, S., 2021. Psychometrics in behavioral software engineering: A methodological introduction with guidelines. ACM Trans. Softw. Eng. Methodol. 31 (1), 1–36.
Grossmann, I., Kross, E., 2010. The impact of culture on adaptive versus maladaptive self-reflection. Psychol. Sci. 21 (8), 1150–1157.
Gudykunst, W.B., 1997. Cultural variability in communication: An introduction. Commun. Res. 24 (4), 327–348.
Gudykunst, W.B., Ting-Toomey, S., 1988. Culture and affective communication. Amer. Behav. Sci. 31 (3), 384–400.
Hall, E., 1976. Beyond Culture. Anchor Books, New York, NY.
Hall, T., Wilson, D., Rainer, A., Jagielska, D., 2007. The neglected technical skill? In Proceedings of the 2007 ACM SIGMIS CPR Conference on Computer Personnel Research: the Global Information Technology Workforce, pp. 196–202. http://dx.doi.org/10.1145/1235000.1235043.
Han, H.-J., 2004. Virtual teams combining mobile devices with web-based communication on group decision making.
Harada, Y., 2017. A cultural comparison of business practices in Thailand and Japan with implications for Malaysia. Cogent Soc. Sci. 3 (1), 1370994.
Herbsleb, J.D., Moitra, D., 2001. Global software development. IEEE Softw. 18 (2), 16–20.
Hofstede, G., 1984. Cultural dimensions in management and planning. Asia Pac. J. Manage. 1 (2), 81–99.
Hofstede, G., 2011. Dimensionalizing cultures: The hofstede model in context. Online Read. Psychol. Cult. 2 (1), 2307–0919.1014.
Hofstede, G., 2016. Culture's consequences: Comparing values, behaviors, institutions, and organizations across nations. Coll. Aviat. Rev. 34 (2), 108.
Hofstede, G., Hofstede, G.J., Minkov, M., 2005. Cultures and Organizations: Software of the Mind. McGraw-Hill New York.
Holmstrom, H., Conchúir, E.Ó., Agerfalk, J., Fitzgerald, B., 2006. Global software development challenges: A case study on temporal, geographical and socio-cultural distance. In: 2006 IEEE International Conference on Global Software Engineering (ICGSE'06), IEEE, pp. 3–11.
Horta, V., Ströele, V., Schettino, V., Oliveira, J., David, J.M., Araújo, M.A.P., 2019. Collaboration analysis in global software development. In: 2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD), IEEE, pp. 464–469.
Inglehart, R., 1981. Post-materialism in an environment of insecurity. Amer. Polit. Sci. Rev. 75 (4), 880–900.
2017. International Telecommunication Union, ICT Facts and Figures [Fact sheet].
Iyer, R., 2019. Effects of Personality Traits and Emotional Factors in Pull Request Acceptance. University of Waterloo.
Jaakkola, H., Heimburger, A., 2009. Cross-cultural software engineering. Informatologia 42 (4), 256–264.
Jaanu, T., Paasivaara, M., Lassenius, C., 2012. Effects of four distances on communication processes in global software projects. In: Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, IEEE, pp. 231–234. http://dx.doi.org/10.1145/2372251.2372293.
Jorgenson, D.W., Vu, K., 2005. Information technology and the world economy. Scand. J. Econ. 107 (4), 631–650.
Kalliamvakou, E., Damian, D., Singer, L., German, D.M., Centric Collaboration, 2014. The code-centric collaboration perspective: Evidence from github. In: The Code-Perspective: Evidence from Github. Technical Report DCS-352-IR, University of Victoria, p. 17.
Kaur, A., Kumar, R., 2015. Comparative analysis of parametric and non-parametric tests. J. Comput. Math. Sci. 6 (6), 336–342.
Kiliç, S., 2016. Chi-square test. Psychiatry Behav. Sci. 6 (3), 180.
Kirkman, B.L., Lowe, K.B., Gibson, C.B., 2006. A quarter century of culture's consequences: A review of empirical research incorporating Hofstede's cultural values framework. J. Int. Bus. Stud. 37 (3), 285–320.
Krippendorff, K., 2018. Content Analysis: An Introduction to its Methodology. Sage publications.
Kunst, J.R., Lefringhausen, K., Skaar, S.W., Obaidi, M., 2021. Who adopts the culture of ethnic minority groups? A personality perspective on majority-group members' acculturation. Int. J. Intercult. Relat. 81, 20–28.
Leidner, D.E., Kayworth, T., 2006. A review of culture in information systems research: Toward a theory of information technology culture conflict. MIS Q. 357–399.
Levene, H., 1961. Robust tests for equality of variances. Contributions to probability statistics. In: Essays in Honor of Harold Hotelling. pp. 279–292.
Li, J., Lam, K., Qian, G., 2001. Does culture affect behavior and performance of firms? The case of joint ventures in China. J. Int. Bus. Stud. 32 (1), 115–131.
Licorish, S.A., 2013. Collaboration Patterns of Successful Globally Distributed Agile Software Teams: The Role of Core Developers. Auckland University of Technology.
Licorish, S.A., MacDonell, S.G., 2013a. Adopting softer approaches in the study of repository data: a comparative analysis. In: Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering, pp. 240–245.
Licorish, S.A., MacDonell, S.G., 2013b. What can developers' messages tell us? a psycholinguistic analysis of jazz teams' attitudes and behavior patterns. In: 2013 22nd Australian Software Engineering Conference, IEEE, pp. 107–116. http://dx.doi.org/10.1109/ASWEC.2013.22.
Licorish, S.A., MacDonell, S.G., 2013c. Self-organising roles in agile globally distributed teams. In: Proceedings of the 24th Australasian Conference on Information Systems (ACIS 2013). pp. 1–10, http://researchbank.rmit.edu.au/view/rmit:161006.
Licorish, S.A., MacDonell, S.G., 2014a. Personality profiles of global software developers. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE 2014). p. 45. http://dx.doi.org/10.1145/2601248.2601265.
Licorish, S.A., MacDonell, S.G., 2014b. Understanding the attitudes, knowledge sharing behaviors and task performance of core developers: A longitudinal study. Inform. Softw. Technol. 56 (12), 1578–1596.
Liu, S.S., Morris, M.W., Talhelm, T., Yang, Q., 2019. Ingroup vigilance in collectivistic cultures. Proc. Natl. Acad. Sci. 116 (29), 14538–14546.
Mairesse, F., Walker, M., 2006. Automatic recognition of personality in conversation. In: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers, pp. 85–88.
Mairesse, F., Walker, M.A., Mehl, M.R., Moore, R.K., 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. J. Artificial Intelligence Res. 30, 457–500.
Mangiafico, S.S., 2016. Introduction to parametric tests.
Mann, L., 1980. Cross-Cultural Studies of Small Groups. Handbook of Cross-Cultural Psychology, Vol. 5. pp. 155–209.
Marinho, M., Luna, A., Beecham, S., 2018. Global software development: practices for cultural differences. In: International Conference on Product-Focused Software Process Improvement, Springer, pp. 299–317. http://dx.doi.org/10.1007/978-3-030-03673-7_22.
McSweeney, B., 2002. Hofstede's model of national cultural differences and their consequences: A triumph of faith-a failure of analysis. Human Relations 55 (1), 89–118.
Nachar, N., 2008. The Mann–Whitney U: A test for assessing whether two independent samples come from the same distribution. Tutor. Quant. Methods Psychol. 4 (1), 13–20.

Nivala, M., Seredko, A., Osborne, T., Hillman, T., 2020. Stack Overflow–Informal learning and the global expansion of professional development and opportunities in programming?. In: 2020 IEEE Global Engineering Education Conference (EDUCON), IEEE, pp. 402–408. http://dx.doi.org/10.1109/EDUCON45650.2020.9125165.
Novielli, N., Calefato, F., Lanubile, F., 2014. Towards discovering the role of emotions in stack overflow. In: Proceedings of the 6th international workshop on social software engineering, pp. 33–36. http://dx.doi.org/10.1145/2661685.2661689.
Oishi, S., Kushlev, K., Benet-Martínez, V., 2021. Culture and personality. Curr. Direct..
Oliveira, N., Andrade, N., Reinecke, K., 2016. Participation differences in Q & A sites across countries: opportunities for cultural adaptation. In: Proceedings of the 9th Nordic Conference on Human-Computer Interaction, pp. 1–10. http://dx.doi.org/10.1145/2971485.2971520.
Oliveira, N., Muller, M., Andrade, N., Reinecke, K., 2018. The exchange in StackExchange: Divergences between Stack Overflow and its culturally diverse participants. In: Proceedings of the ACM on Human–computer Interaction, 2 (CSCW), pp. 1–22.
Ouriques, R.A.B., Wnuk, K., Gorschek, T., Svensson, R.B., 2019. Knowledge management strategies and processes in agile software development: a systematic literature review. Int. J. Softw. Eng. Knowl. Eng. 29 (03), 345–380.
Oyserman, D., 2017. Culture three ways: Culture and subcultures within countries. Ann. Rev. Psychol. 68, 435–463.
Öztuna, D., Elhan, A.H., Tüccar, E., 2006. Investigation of four different normality tests in terms of type 1 error rate and power under different distributions. Turkish J. Med. Sci. 36 (3), 171–176.
Patel, D., 2017. Key to success of offshore outsourcing.
Patrick, H.A., Kumar, V.R., 2012. Managing workplace diversity: Issues and challenges. SAGE Open.
Pennebaker, J.W., King, L.A., 1999. Linguistic styles: language use as an individual difference. J. Personal. Soc. Psychol. 77 (6), 1296.
Pennebaker, J.W., Lay, T.C., 2002. Language use and personality during crises: Analyses of Mayor Rudolph Giuliani's press conferences. J. Res. Personal. 36 (3), 271–282.
Pennebaker, J.W., Mehl, M.R., Niederhoffer, K.G., 2003. Psychological aspects of natural language use: Our words, our selves. Ann. Rev. Psychol. 54 (1), 547–577.
Potts, C., 1993. Software-engineering research revisited. IEEE Softw. 10 (5), 19–28.
Rahman, A.U., Khan, K., Kamal, S.W., Naveed, H., Bacha, M., 2020. Use of collaborative tools and modern technologies as critical success factor in global software development. I-Manager's J. Softw. Eng. 15 (1), 48.
Rashid, M., Clarke, P.M., O'Connor, R.V., 2019. A systematic examination of knowledge loss in open source software projects. Int. J. Inf. Manage. 46, 104–123.
Rastogi, A., Nagappan, N., 2016. On the personality traits of github contributors. In: 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), IEEE, pp. 77–86.
Rice, R.E., D'Ambra, J., More, E., 1998. Cross-cultural comparison of organizational media evaluation and choice. J. Commun. 48 (3), 3–26.
Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14 (2), 131–164.
Salkind, N.J., 2006. Encyclopedia of Measurement and Statistics. SAGE Publications.
Schenk, D., Lungu, M., 2013. Geo-locating the knowledge transfer in StackOverflow. In: Proceedings of the 2013 international workshop on social software engineering, pp. 21–24. http://dx.doi.org/10.1145/2501535.2501540.
Schwartz, S., 2006. A theory of cultural value orientations: explication and applications. Compar. Sociol. 5 (2–3), 137–182.
Selvi, A.F., 2019. Qualitative Content Analysis. Routledge, pp. 440–452.
Shneiderman, B., 1980. Natural vs. precise concise languages for human operation of computers: Research issues and experimental approaches. In: 18th Annual Meeting of the Association for Computational Linguistics, pp. 139–141.
Smite, D., Moe, N.B., Levinta, G., Floryan, M., 2019. Spotify guilds: how to succeed with knowledge sharing in large-scale agile organizations. IEEE Softw. 36 (2), 51–57.
Sowe, S.K., Stamelos, I., Angelis, L., 2008. Understanding knowledge sharing activities in free/open source software projects: An empirical study. J. Syst. Softw. 81 (3), 431–446.
Steiber, A., Alänge, S., 2013. A corporate system for continuous innovation: the case of google inc. Eur. J. Innov. Manage..
Stevenson, A., 2010. Oxford Dictionary of English. Oxford University Press, USA.
Tausczik, Y.R., Pennebaker, J.W., 2010. The psychological meaning of words: LIWC and computerized text analysis methods. J. Language Soc. Psychol. 29 (1), 24–54.
Teeni, D., 2001. A cognitive-affective model of organizational communication for designing IT. MIS Q. 251–312.
Tian, Y., Ng, W., Cao, J., McIntosh, S., 2019. Geek talents: Who are the top experts on github and stack overflow? CMC-Comput. Mater. Continua 61 (2), 465–479.
Tjosvold, D., Law, K.S., Sun, H.F., 2003. Collectivistic and individualistic values: Their effects on group dynamics and productivity in China. Group Decis. Negot. 12 (3), 243–263.
Triandis, H.C., 1998. Vertical and horizontal individualism and collectivism: Theory and research implications for international comparative management. Adv. Int. Compar. Manage. 12, 7–36.
Tukey, J., 1977. Exploratory Data Analysis. Addison-Wesley, Reading, MA. -.
Vallaster, C., 2005. Cultural diversity and its impact on social interactive processes: Implications from an empirical study. Int. J. Cross Cultural Manage. 5 (2), 139–163.
Van Den Hooff, B., De Ridder, J.A., 2004. Knowledge sharing in context: the influence of organizational commitment, communication climate and CMC use on knowledge sharing. J. Knowl. Manage..
Vargha, A., Delaney, H.D., 1998. The Kruskal–wallis test and stochastic homogeneity. J. Educ. Behav. Stat. 23 (2), 170–192.
Vasilescu, B., Filkov, V., Serebrenik, A., 2015. Perceptions of diversity on git hub: A user survey. In: 2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering, IEEE, pp. 50–56. http://dx.doi.org/10.1109/CHASE.2015.14.
Watkins, H.S., Liu, R., 1996. Collectivism, individualism and in-group membership: implications for consumer complaining behaviors in multicultural contexts. J. Int. Consumer Market. 8 (3–4), 69–96.
Weinberg, G.M., 1972. Programming and compiling strategies for paging systems. Softw. Pract. Exp. 2 (2), 165–171.
Weisstein, E.W., 2004. Bonferroni correction.
Wickberg, D., 2007. The sympathetic self in American culture: 1750-1920. In: Figures in the Carpet: Finding the Human Person in the American Past. pp. 129–161.
Wikfeldt, E., 2016. Generalising from case studies.
Wiklund, K., Eldh, S., Sundmark, D., Lundqvist, K., 2017. Impediments for software test automation: A systematic literature review. Software testing. Verif. Reliab. 27 (8), e1639.
Wilczewski, M., Gut, A., Gorbaniuk, O., 2017. The impact of individualism-collectivism orientation and communal orientation on employees' attitudes toward intercultural communication: The case of chinese employees in an MNC. J. Intercult. Commun. 45, 106–131.
Wohlrab, R., Knauss, E., Steghöfer, J.-P., Maro, S., Anjorin, A., Pelliccione, P., 2020. Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture. Requir. Eng. 25 (1), 21–45.
Yao, J., Crupi, A., Di Minin, A., Zhang, X., 2020. Knowledge sharing and technological innovation capabilities of Chinese software SMEs. J. Knowl. Manage..
Yee, N., Harris, H., Jabon, M., Bailenson, J.N., 2011. The expression of personality in virtual worlds. Soc. Psychol. Personal. Sci. 2 (1), 5–12.
Yin, R.K., 2009. Case Study Research: Design and Methods. SAGE Publications, sage, Thousand Oaks, CA.
Zhou, Y., Kwon, J.-W., 2020. Overview of hofstede-inspired research over the past 40 years: The network diversity perspective. SAGE Open 10 (3), 2158244020947425.
Zhu, E., 1996. In: Simonson, M.R., et al. (Eds.), Meaning Negotiation, Knowledge Construction, and Mentoring in a Distance Learning Course. Washington, DC, pp. 821–844.
Zolduoarrati, E., Licorish, S.A., 2021. On the value of encouraging gender tolerance and inclusiveness in software engineering communities. Inf. Softw. Technol. 139. http://dx.doi.org/10.1016/j.infsof.2021.106667.

Elijah Zolduoarrati is a Research Assistant and Ph.D. student in the Department of Information Science at the University of Otago, New Zealand, where he also earned his BAppSc and M.Sc. His research portfolio covers applied data science, big data, bioinformatics, cloud computing, computational linguistics, cyber security, data (analysis, architecture, brokerage, cleansing, exploration, inspection, management, mining, modelling, predictions, preservation, reproducibility, transformation and visualisation), diversity (gender inclusivity and equality), education, human–computer interaction, human factors, information assurance, internet of things, machine learning, micromarketing, operating systems, quantitative and qualitative research methods and techniques, robotics, social and behavioural sciences also software engineering.

Sherlock A. Licorish is a Senior Lecturer in the Department of Information Science at University of Otago, New Zealand. He was awarded his Ph.D. by Auckland University of Technology (AUT), and joined University of Otago in 2014. His research portfolio covers agile methodologies, practices and processes, teams and human factors, human–computer interaction, research methods and techniques, software code quality, static analysis tools, machine learning applications and software analytics. Sherlock occupies several service roles across the university, nationally and internationally.

Nigel Stanger is a Lecturer in the Department of Information Science at the University of Otago, New Zealand, where he also earned his Ph.D. His research interests include applied data science (focusing particularly on the software engineering domain), data analysis reproducibility, data management and preservation, human factors in software engineering, defect prediction, and computing education. He has served on the committees of EASE, eScience, ICCE, and APCCM, and was the general co-chair of the 2021 Australasian Computer Science Week multi-conference. He is a member of ACM and IT Professionals New Zealand.