Towards Integrated Dashboards for Better Management of Human-centric Issues in Software Development

Liam Todd[1], Kashumi Madampe[1*], Hourieh Khalajzadeh[2], Mojtaba Shahin[3], John Grundy[1]

[1]HumaniSE Lab, Monash University, Clayton, Victoria, Australia.
[2]School of Information Technology, Deakin University, Melbourne, Australia.
[3]School of Computing Technologies, RMIT University, Melbourne, Australia.

*Corresponding author(s). E-mail(s): kashumi.madampe@monash.edu;
Contributing authors: ltod0001@student.monash.edu;
hkhalajzadeh@deakin.edu.au; mojtaba.shahin@rmit.edu.au;
john.grundy@monash.edu;

Abstract

GitHub and Jira projects typically contain many issues and issue comments used to track project tasks and defects. An important class of issues that needs appropriate consideration is called *"human-centric issues"*. These issues relate to different human characteristics of end users that need to be identified, tracked and managed differently from traditional technical-related issues. Current management of these human-centric issues during defect management is limited. We introduce a novel dashboard – the (Human-centric Issue Visualiser – HCIV) that categorises and tags these HCIss. We built HCIV prototypes for the two platforms, GitHub and Jira. These tag issues and present them in various visual forms to software practitioners. Using the dashboard, human-centric issues can be prioritised and tracked, and machine learning-generated classifications can be overridden. To reflect these interactions, associated GitHuband Jira issue tags are updated while the user interacts with our dashboard. The user evaluations of our dashboard prototypes show their potential for human-centric issue management. A demo of the GitHub version of the tool being used can be viewed at https://youtu.be/v49aiRiDIPs, and the Jira version can be viewed at https://youtu.be/qQM72SErmqs.

Keywords: GitHub, issue comments, visualisation, human-centric issues

1 Introduction

Software systems are used by diverse groups of end-users and are designed to fulfill their expectations. However, software solutions may be hard-to-deploy, hard-to-maintain, and hard-to-use if *human-centric issues* (HCIs) are neglected during the software development process [1–5]. HCIs are defined as *"the problems end-users might face when using software that stem from the lack of (proper) consideration of their specific/differing human characteristics, limitations, abilities, personalities, technical proficiency, emotional reactions to software systems, gender, age, and so on"* [6]. However, many software developers are not explicitly conscious of such issues, nor have they experienced them. Furthermore, many do not understand and effectively communicate the implications of HCIs [7].

Our recent work investigated the phenomenon of HCI discussion in app reviews and GitHub repositories [6]. We proposed a taxonomy of these HCIs, including three high-level categories: *App Usage*, *Inclusiveness*, and *User Reaction*. The App Usage category is made up of issues related to *Resource Usage*, *Buginess*, *Change & Update*, *UI & UX*, *Privacy & security*, *Usage instruction*, *Access issues*, and *Others*. The Inclusiveness category comprises issues related to *Compatibility*, *Location*, *Language*, *Accessibility*, and *Others*. Lastly, the User Reaction category includes issues related to *Fulfilling interests*, *Emotional aspects*, *Preference*, and *Others*. We also investigated the use of machine learning-based classifiers that can identify with high accuracy such HCIs [6, 8, 9]. However, these proposed classifiers generally have two key limitations: (1) they are stand-alone programs that can not directly aid developers when maintaining their software and addressing its human-centric defects; and (2) they do not provide any effective visualisations of HCIs identified in a GitHub repository.

We wanted to develop a proof-of-concept dashboard that supports software practitioners in better considering and managing these HCIs throughout the software development life cycle. We built two dashboard prototypes – the Human-centric Issue Visualiser (HCIV) – for GitHub and Jira. The dashboard uses a machine learning classifier to automatically tag all HCIs identified in a GitHub repository and a Jira project, respectively. Via the dashboard, the identified HCIs can be prioritised, tracked, and monitored for progress while fixing these defects. We carried out two user studies for both GitHub and Jira versions of the tool that demonstrate the promise of our proposed HCI dashboard approach.

The rest of this paper is organised as follows. Section 2 discusses the motivation and background of this study. Next, we present the architecture of the two prototypes in Section 3, while Section 4 presents the UI designs and usage examples and Section 5 presents the evaluations of the prototypes. Next, we present the discussion in Section 6 followed by the summary of our work in Section 7.

2 Motivation and Background

2.1 Motivation

Human aspects – including *age*, *gender*, *culture*, *language and location*, *digital literacy*, *physical and mental impairments*, and *differing personalities and preferences* – have a strong influence on the adoption of software [10]. Their negligence brings about significant implications [6, 9]. Consider a dyslexic individual who wishes to acquire information about their diet from a website. This user has specific requirements in order to access the content on the website. As one of the most widely used software repositories, GitHub provides end-users and developers with the ability to provide feedback on a software system (e.g., a diet website) via its issue tracker, if it is hosted on GitHub. GitHub issue discussions in the issue tracker are initiated with a title (*issue title*), and are followed by comments (*issue comments*) from the reporter and contributors, who may be developers, users, project maintainers, or the reporter themselves.

Such an issue in GitHub's issue tracking system, opened by a collaborator to discuss the dyslexic user's requirements, is shown in Figure 1. A follow-up comment is also shown, in which a collaborator lists further techniques to support individuals with dyslexia in their use of the website, and makes suggestions for the website such that it is more inclusive of its diverse end users.

This scenario exemplifies the value of attending to, and actively discussing issues related to human aspects (i.e., we refer to such issues in this paper as *HCIs*) in the uptake of software. However, there is currently no way to tag and manage such issues systematically. We wanted to use a machine learning classifier to automatically add suitable tags to GitHub issues to identify different classes of HCIs in a GitHub project. We wanted to give managers and team members a set of visualisations to be able to see a high level overview of different HCIs, prioritise and assign them to appropriate developers, and track their fixing progress over time.

To the best of our knowledge, to auto-tag Trello cards or Jira tickets, the user has to set some rules. In our case, we mitigate that by running the ML model in the backend. We also found when evaluating our human-centric classifier [6] that built-in issue management support e.g. searching for issues with specific human-centric issue tags, could provide developers with limited support for some human-centric issue management. However, this support was limited which motivated the exploration of visualising and managing human-centric issues in GitHub via a stand-alone tool – our GitHub Human-centric Issues Visualiser (HCIV) prototype. This has the major disadvantage of being a separate tool, and this motivated our integrated Jira HCIV prototype.

2.2 Motivating Usage Examples

Below we present how our HCIV concept can be put into practice using a couple of representative usage examples.

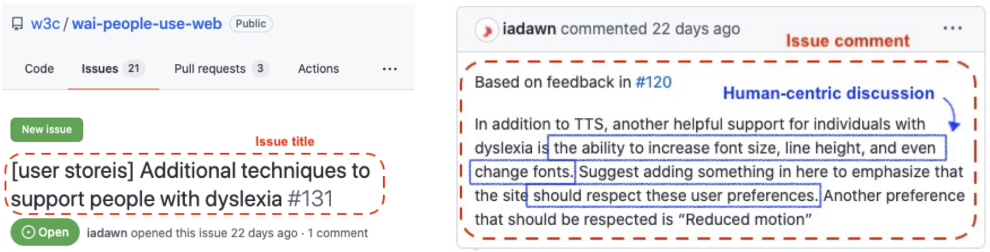

Fig. 1 Example HCI (left) and follow-up comment (right) from GitHub (from [6])

2.2.1 Managing Human-centric Issues

Consider a user of the application, Xiaoyu, who lives in a Southeast Asian country and is a software developer by profession. Xiaoyu notices a few problems regarding the application's inclusiveness of its diverse user base: the application lacks currency conversions to many Southeast Asian currencies, and the in-app self-guided tour is only offered in English. Xiaoyu knows that the application is open source. So, Xiaoyu decides to add some GitHub issues to its repository in hopes that a contributor will address them. An active contributor to the application, Yun, decides to link the application's repository to HCIV. Upon inspection of the 'HCI Categorisation Count' plot on the Overview View, Yun notices that there are disproportionately many inclusiveness-related issues in the repository. Yun uses the Prioritise View to assign these issues with 'high' priority due to the clear evidence that the inclusiveness of the application is lacking in comparison to the other HCI categories. Another active contributor to the project, Zhipeng, checks the issues in the GitHub repository with the intention of taking responsibility for an issue and raising a pull request that resolves it. Noticing that a few issues have been labelled with 'inclusiveness' and 'high priority', Zhipeng picks up one of these issues and promptly resolves it.

From this example, we can see how only one individual needed to interact with HCIV for the human-centric aspects of the project to be improved upon, as persisting yet unobtrusive changes were made to the GitHub repository through the tagging of issues. It is also worth noting that direct communication did not occur between any of the individuals in the scenario. Note that for HCIV to have a meaningful impact on the project, GitHub Issues must be actively used for bug-reporting, feature requests, hosting discussions, processing support requests, and so on.

2.2.2 Agile Team Lead Management

Now consider an example of Team Lead, Yanjie, whose Agile team is working on a desktop application for video editing and has just finished their third sprint. Yanjie links the GitHub repository containing the project's code to HCIV with the intention of improving the team's efficiency for future sprints, because the previous sprint seemed to be far more time-crammed than the first two despite the sprint planning sessions involving the inclusion of the same number of new features. Upon inspection of the 'HCIs Against Time' plot, Yanjie notices that there is a spike of user reaction-related

issues on the days following the 10th of October - the beginning of the third sprint. This paints a clear picture for Yanjie, as it was only after the second sprint that a product release occurred, and thus, customer feedback started to be heard and discussed within the Agile team.

From the information presented by HCIV, as well as the context surrounding the project, Yanjie deduces the cause of the spike of user-reaction related issues at the beginning of the third sprint: Prior to the release at the conclusion of sprint 2, a lack of consideration was made about what type of media would be edited using the product - the team had assumed that users would be editing short to medium length, cut-heavy videos and failed to consider the use cases for podcast editing. This version of the product did not fulfil the interests of a large portion of its end users, resulting in many user-reaction related issues being subsequently raised in the GitHub repository, unexpectedly increasing the team's workload during the third sprint. This observation enables Yanjie to improve the team's processes, by dedicating time during sprint planning sessions to consider the specific preferences, interests, and emotional reactions of the end users with regard to the previous release of the product. By undertaking such an exercise to generate issues, through the use of personas, for example, some of the team's capacity may be used to address these issues at the onset of the sprint, reducing the likelihood of new issues being brought up during sprints, originating from user feedback, spreading the team's capacity too thinly. This example demonstrates how HCIV's data visualisations may be used in an Agile environment to help improve a team's processes.

2.3 Related Work

Various platforms including online repositories, question and answer websites, and issue tracking platforms such as GitHub, Stack Overflow, and Jira have an abundance of data about the technical side of software development processes. They also include valuable information which can provide insights into the **social and human aspects** of the software development process [11] and various human aspects of end users of the software [6, 9].

Due to its numerous open source projects and the myriad of technical and non-technical information to be mined, software engineering researchers have been experimenting with extensions to GitHub for many years [12]. Much work has been done on mining technical and human aspects of software from GitHub and similar repositories. For example, Ko et al. [13] analysed design discussions between developers in Bugzilla bug reports to attain an understanding of the design challenges and how decisions are made to adapt to the needs of end users. Several works have analysed usability bug reports in Bugzilla and developed classifications from them [14, 15]. Andreasen et al. mined software repositories and used surveys and interviews to investigate developers' opinions regarding usability [16]. Pleta et al. mined from discussions around commits and pull requests in their exploration of GitHub discussions relating to security [17]. There have also been studies which mined social aspects in repositories. Dabbish et al. mined GitHub for transparency and collaboration in GitHub projects [18], and Dam et al. mined open-source projects to discover social norms

[19]. Novielli et al. conducted an empirical study on affective lexicons in the questions posted on Stack Overflow. This study shows that affective lexicons in technical questions influence the likelihood of procuring a satisfying response to said questions [20].

Cabrera-Diego et al. developed classifiers for comments related to emotions on StackOverflow and Jira [21]. Khalajzadeh et al. [9] conducted an empirical study to classify HCIs into eight categories: Inclusiveness, Privacy & Security, Compatibility, Location & Language, Preference, Satisfaction, Emotional Aspects, and Accessibility. This was achieved through the manual analysis of 1,691 issue comments from 12 diverse projects, ranging from small to large in their scale. Nurwidyantoro et al. investigated a way to help address human values during software development, through the use of a human values dashboard [22]. This study produced a set of high-level requirements for a human values dashboard, many of which were mapped to an equivalent high-level requirement for the HCI dashboard presented in this paper. Barcellini et al. analysed and visualised social, thematic temporal, and design aspects of online software repositories to understand and model the dynamics of the open source software design process in mailing list exchanges [23].

2.4 Our Approach

We wanted to provide a range of useful visualisations to developers for automatically classified HCIs reported on GitHub and Jira with the assistance of a machine-learning classifier. Users can override these automatic classifications, prioritise them, and organise project tasks around them. Our dashboards provide charts summarising the change in the number and type of HCIs over time in a GitHub repository, display the distribution of issues among various HCI categories in a repository and a Jira project, and support tracking of the priority and progress of HCIs for a GitHub repository and a Jira project. We call our dashboard a Human-Centric Issue Visualiser (HCIV).

In summary, the GitHub version of HCIV allows users to view summaries of HCIs in the connected project, view list and issue details of each issue, modify issue fields, view issues grouped by priority and progress, and move issues between priority and progress groups. The Jira version of HCIV allows users to filter issues and view key statistics of the issues of the connected Jira project, view classification distribution of the issues, view issues grouped by priority and status, view the issue list and the HCI categories, and view the issue panel. These features are explained in the next sections of the paper.

3 Architecture

3.1 Architecture of the GitHub-integrated Prototype

We first developed a GitHub-integrated prototype dashboard for HCIV. This can be linked to a GitHub repository, the issues of which are automatically categorised into four categories via a machine-learning classifier: User Reaction, App Usage, Inclusiveness, and Non-human centric issues. A range of different visualisations are provided by the dashboard, including a number of data visualisations to gain a big-picture view

of the repository in terms of its HCIs; a filterable list of the issues with detailed information about each; a progress tracking tool; a prioritisation tool; a HCI categorisation correction tool; and a tool to remove all changes to the targeted repository made by the prototype. We used GitHub labels to ensure that the information available through HCIV would be visible and persistent on GitHub.

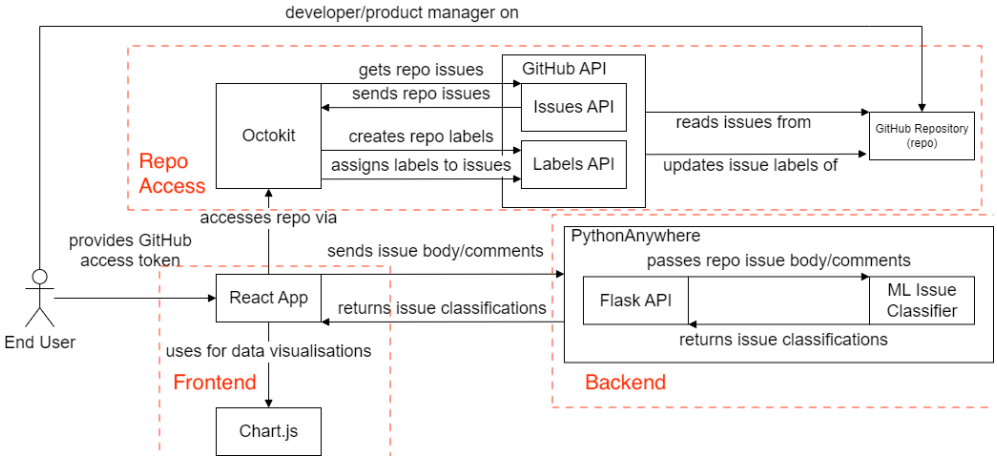

Fig. 2 Outline of the architecture of our GitHub human-centric issue visualiser prototype.

GitHub Prototype. Fig. 2 shows an outline of the architecture of our GitHub human-centric issue visualiser prototype. HCIV is composed of a user-facing web application hosted on GitHub Pages - the frontend, and a cloud-hosted REST API providing access to our machine-learning classifier [6] - the backend. The frontend communicates with the backend and a target GitHub repository. The frontend web application is built with the popular JavaScript library, React, due to its flexibility and reusability, along with rich documentation and support. The user performs queries and manipulations on the target GitHub repository linked to HCIV via this user interface. This integration is achieved using Octokit, a GitHub-supported client library for the GitHub API. This library is used to interact with the GitHub API, communicating with the target user-selected GitHub repository to query issues, create and update tags, and add/update/delete tags. The frontend uses the Chart.js library to implement its HCI data visualisations. React-chartjs-2, a JavaScript library that provides React components for Chart.js library, was utilised to implement a variety of chart visualisations. The HCIV frontend web application is hosted on GitHub pages for fast and easy deployment and an easy point of access.

Human-centric Issue Classifier. We wanted to use a machine learning model to simplify the process of packaging the classifier for integration with our frontend. We selected the highest score machine learning model developed by [6]. This is a combination of [Classifier chain + TF-IDF (analyser = char, ngram = 4,4)] with

base Linear SVM performance (Accuracy-0.786, F1 score-0.797, and Hamming loss-0.104). We used the pickle module to serialise and package the classifier that was trained on 1,200 GitHub issue comments in this work [6] and then de-serialise and use it to pre-process and classify our GitHub issue comments. Flask, a lightweight Python web framework, was used to implement a REST API to provide the frontend with access to our machine-learning classifier. We chose a Python framework due to various Python-specific libraries being needed to interact with the machine-learning classifier. Flask was the chosen framework because it is more lightweight compared to an alternative like Django, and is supported by many cloud-hosting services. The REST API is hosted using PythonAnywhere, a Python-specific cloud-hosting service. The API serves a single endpoint which returns the HCI classification of a GitHub issue comment, and is publicly available. Reusability was emphasised as our machine-learning classifier is non-specific to HCIV. Implementing the backend as a publicly available API may save development time in future projects which require its novel capabilities. The source code github repository for this initial visualiser can be found at: https://github.com/LiamTodd/human-centric-issue-visualiser

3.2 Architecture of Jira-integrated Prototype

After evaluating our GitHub integrated prototype, user feedback indicated closer integration of the issue tracking system and human-centric issue visualisation would be a better approach. To this end, we developed a Jira-based prorotype, an architecture diagram for which is shown in Fig. 3. As an integrated Jira plug-in this makes use of Jira features to realise front end and repository integration. The tool consists of a Forge App and a Flask app. The frontend of the Forge app includes the human-centric issue dashboard and an issue-panel visualisation. These were both built as two React apps. These two React apps then make use the Jira REST API to communicate to retrieve and update the Jira issues in the Atlassian Ecosystem repository. The Forge app consists of a backend with Forge resolver functions, which retrieve issue classifications from the Flask app. The Flask app classifies issues using our ML issue classifier, explained in the previous section. It uses our classifier API by sending it issues to classify and receiving a human-centric issue classification for recording in Jira in response. The JIRA prototype source code can be found at: https://github.com/LiamTodd/Jira-HCIV

4 UI Designs and Usage Examples

4.1 Key Design Decisions

In our human-centric issue visualisation tools, we wanted to support key tasks of:

- have all existing project issues classified using our human-centric classifier
- get an overview of the existing human-centric issues in their project
- prioritise human-centric issues for resolution
- track progress at resolving human-centric issues
- comment on and resolve human-centric issues

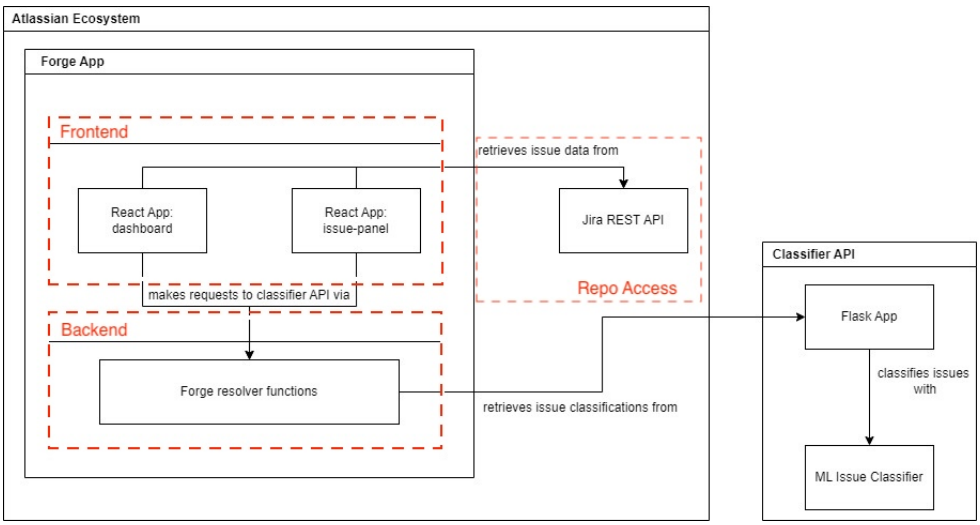

Fig. 3 The architecture of HCIV – Jira version (The tool consists of a Forge App and a Flask app. The Forge app consists of a backend with Forge resolver functions, which retrieve issue classifications from the Flask app. The Flask app classifies issues using our ML issue classifier)

Initially in our GitHub prototype we chose to visualise charts capturing key information using charts provided by the React-chart-2.js library for simplicity. These provide a range of React web interface charts with minimal effort, supporting our prototyping efforts. We chose basic chart renderings and colour scheme, expecting to refine these based on user feedback. For our Jira prototype, we used user feedback from the evaluation of our GitHub prototype's interfaces to modify some of the visulisations. We also used Jira-provided chart support to provide a more integrated look and feel with other Jira-based visualisations.

4.2 GitHub Prototype UX and Usage Examples

The developer specifies a target GitHub repository

The starting point of HCIV is where the developer starts by specifying a target GitHub repository they have the right to update. HCIV accesses the GitHub repository, extracts all issues, runs our machine learning classifier over them, and then generates tags which it adds to issues via the GitHub API. A list of example issues with HCIV-generated tags is shown in Figure 4. The progress and priority tags are generated by the progress and priority views in HCIV, described below.

Developer views summaries of human-centric issues in the connected GitHub project

A variety of high-level views are provided that can present a summary of all HCIs, or a filtered set via several criteria (HCI category, priority, progress, etc). We chose these chart visualisations to provide simple summary views for users and utlised wherever possible existing React-chart-2.js chart support. This constrained some look and feel

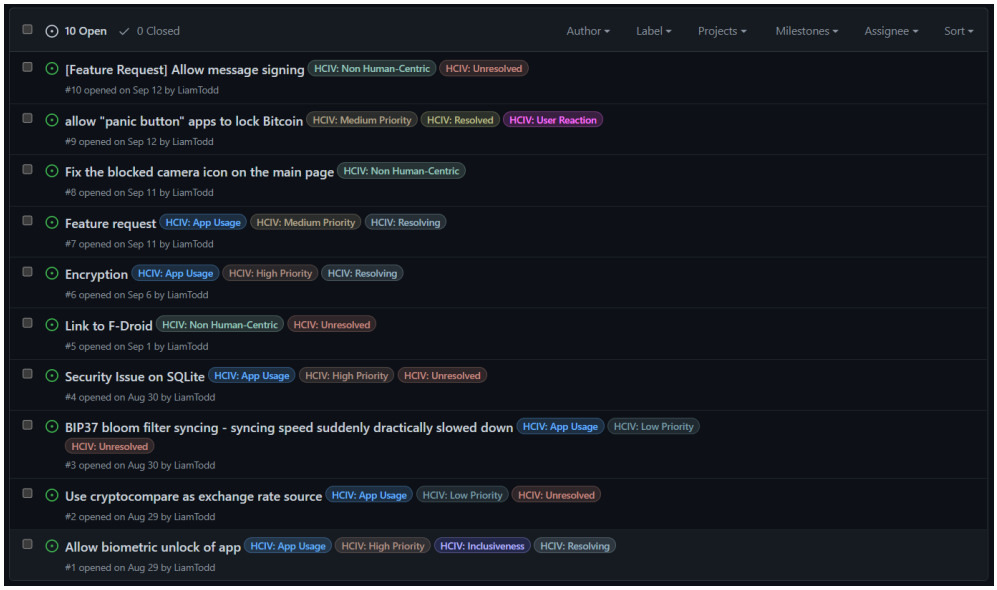

Fig. 4 Example of a tagged set of issues in a repo

aspects of the visualisations but allowed rapid prototyping and evaluation. A timeline bar chart is shown in Figure 5 showing the change of issue count/issue type over time. This is used by the developer to track human-centric issue occurrence and resolution over specified timelines. The example shows a number of human-centric issues occurring and being resolved between Aug 29 and Sept 01, then a number more occurring Sept 11 and 12. The design rationale was to provide a timeline representation for high-level human-centric issue change in a project.

A categorisation polar area chart showing the four types of tagged HCIs is shown in Figure 6. This is used by the developer to see the different kinds of human-centric issues currently tagged in the repository. The example shows a large number of app usage issues in the example repository, and only one each of user reaction and inclusiveness human-centric issues. The design rationale was to provide a comparison of occurance of different kinds of human-centric in a subset of selected issues.

Figure 7 shows a summary of 'issue progress' i.e. progress towards completing the fixing of a reported human-centric defect. This is useful for managers to track issue resolution, and again can be filtered in various ways to focus on a subset of issues in the target project. The design rationale was to provide a concrete representation of relative progress of addressing human-centric issues impacting a subset of selected issues.

Figure 8 shows a summary of issue priority. HCIV allows managers to tag HCIs with a 'priority' for fixing (described below). This is useful for developers to see the relative incidence of different developer-assigned priority levels for the current instances of human-centric issues currently tagged in the repository. The example shows a fairly even mix of different priority human-centric issues present.

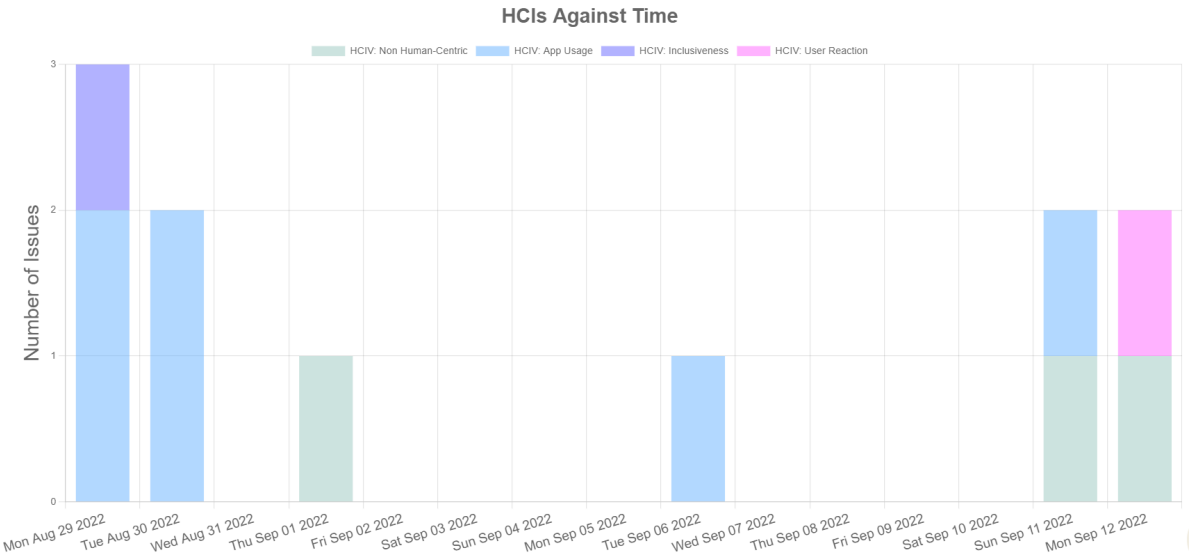

Fig. 5 Human-centric issues over time bar chart example (The change of issue count/issue type over time)

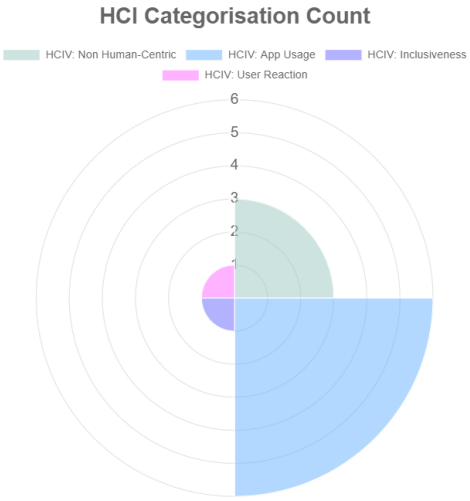

Fig. 6 Visualising human-centric issue classification distribution

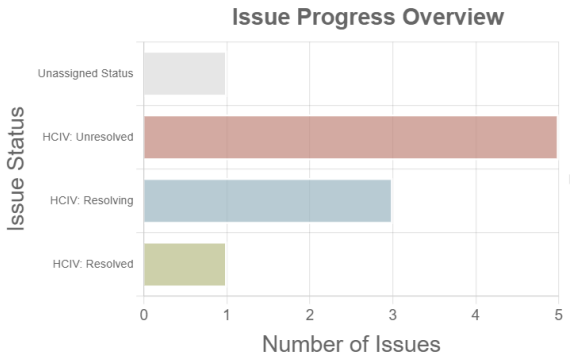

Fig. 7 Visualising progress summary (The progress towards completing the fixing of a reported human-centric defect)

Developer views HCIs grouped by priority and grouped by progress

Two views of grouped HCIs are provided, illustrated in Figure 9 and Figure 10. The priority view in Figure 9 shows issues grouped by 'priority'. As above, we wanted to

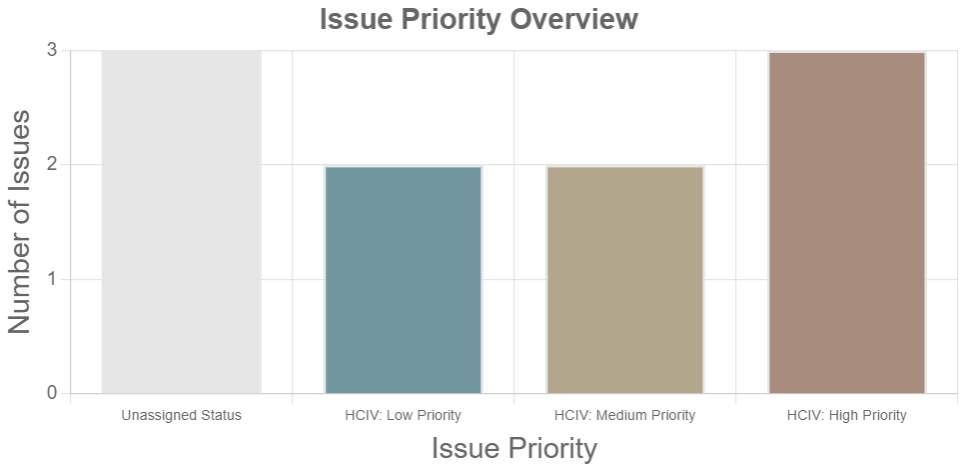

Fig. 8 Visualising issue priority summary

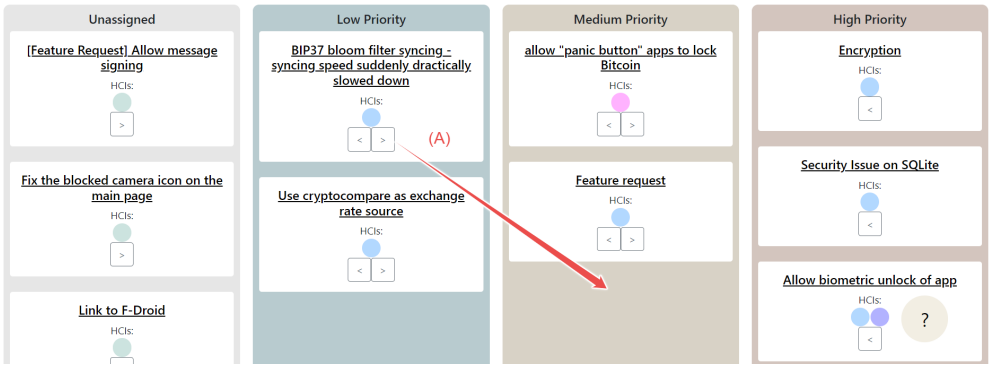

Fig. 9 Interactive change priority view

provide an easy to use motification interface within the visualisation tool, using the same colour and tagging scheme as the various chart visualisations.

Developer moves HCIs between priority and progress groups

Initially, all issues are assigned to the 'unassigned' category. The developer can use arrow buttons to move issues between priority groups. For example, Figure 9 (A) shows an example of an issue priority being prompted by drag and drop. This results in its GitHub repository issue tag being updated via the GitHub API. A set of issues grouped on 'progress' is shown in Figure 10. The progress of an issue can similarly be updated by using arrow buttons, as you can see in Figure 10 (A), also resulting in the issue's GitHub repository tag being changed via the GitHub API.

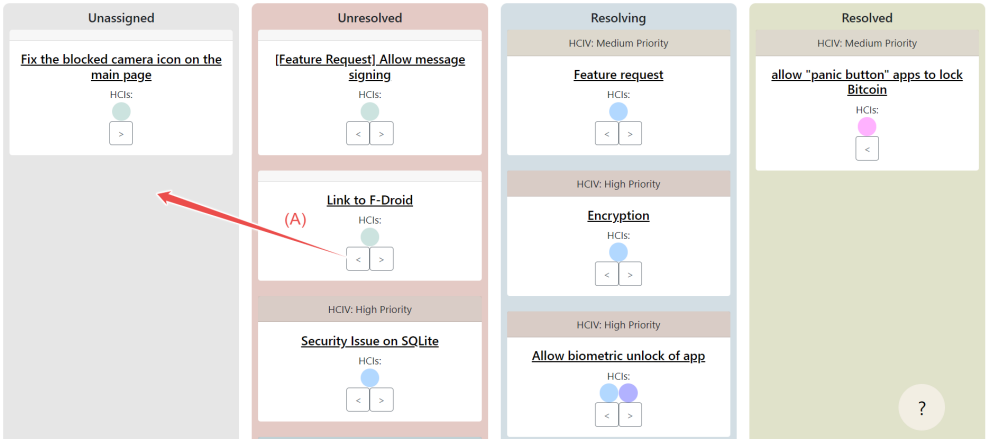

Fig. 10 Interactive change progress view

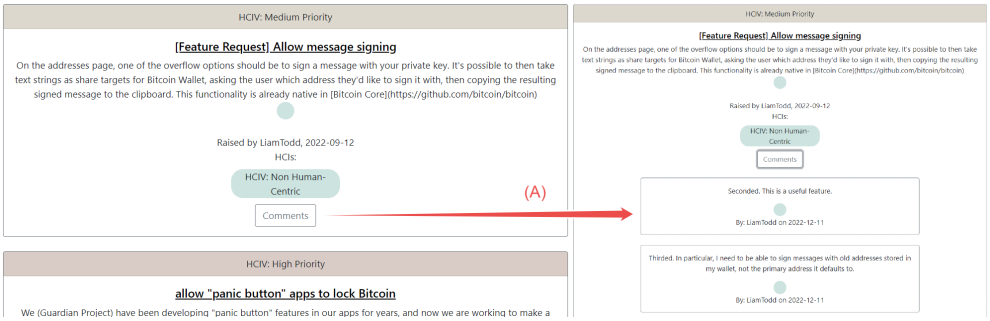

Fig. 11 List of issues (left), issue details (right)

Developer views list and issue panel (details) of each issue

HCIV allows details of each human-centric tagged issue to be viewed, including all tags, owner, dates, and comments. Figure 11 (left) shows an example of an issue list view, also filterable on several issue criteria. The developer can select an issue and expand its comments (Figure 11 (right) (A). We wanted to represent an issue summary and provide interactive modification of human-centric properties within our issue visualisation tool vs forcing the user to use GitHub's built in tag and comment editor within GitHub. However, these native GitHub issue editing mechanisms can also be used.

Developer modifies defect issue fields

HCIV allows the developer to modify various fields including HCI classification tags – add or remove; priority and progress tags – add, remove, and change (Figure 12 (A). Modifying the automatically generated HCI tags allows the developer to override

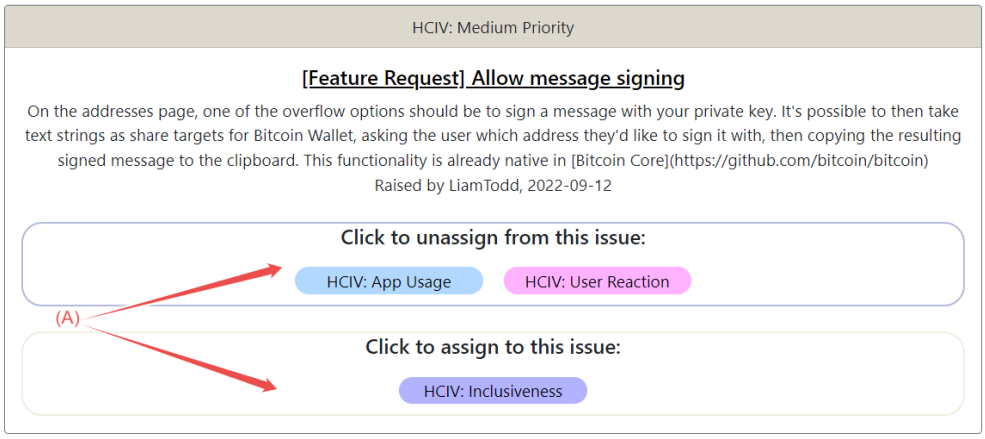

Fig. 12 Correcting human-centric issue tags

our machine learning classifier tagging when it makes a mistake. Setting priority and progress tags supports HCI defect fixing project management.

4.3 Jira Prototype UX and Usage Examples

Developer filters issues and views the key statistics of the issues

Fig. 13 shows the dashboard filters and key statistics views of the HCIV Jira plugin. The filter allows the developer to filter the dashboard to show/hide non-human centric issues and filter by date created (in the form of a range), by status, and by priority. The key statistics section of the dashboard shows the statistics: percentage of the issues that are human-centric and the assigned number of HCIs according to the members of the project. The key statistics section can also be expanded, which provides the issues and the counts/percentages according to the HCI category. Here can be seen the integration of our visualisations within Jira itself as opposed to a separate React-based web interface as used in the GitHub UX. Using the Jira issue filtering directly, the developer selects issues based on a variety of criteria and can display various aggregate charts of specific human-centric issue properties within Jira's web interface. The statistics visualisation was motivated from user feedback from our GitHub user evaluation.

Developer views HCI classification distribution of the issues in the Jira project

Fig. 14 shows the HCI classification distribution view as realised in our Jira prototype. As in the GitHub prorotype, this allows the developer to view the HCI category distribution by overall issues, summary, and descriptions. The view also allows the developer to switch between the bar-chart and pie-chart modes. Unlike the GitHub

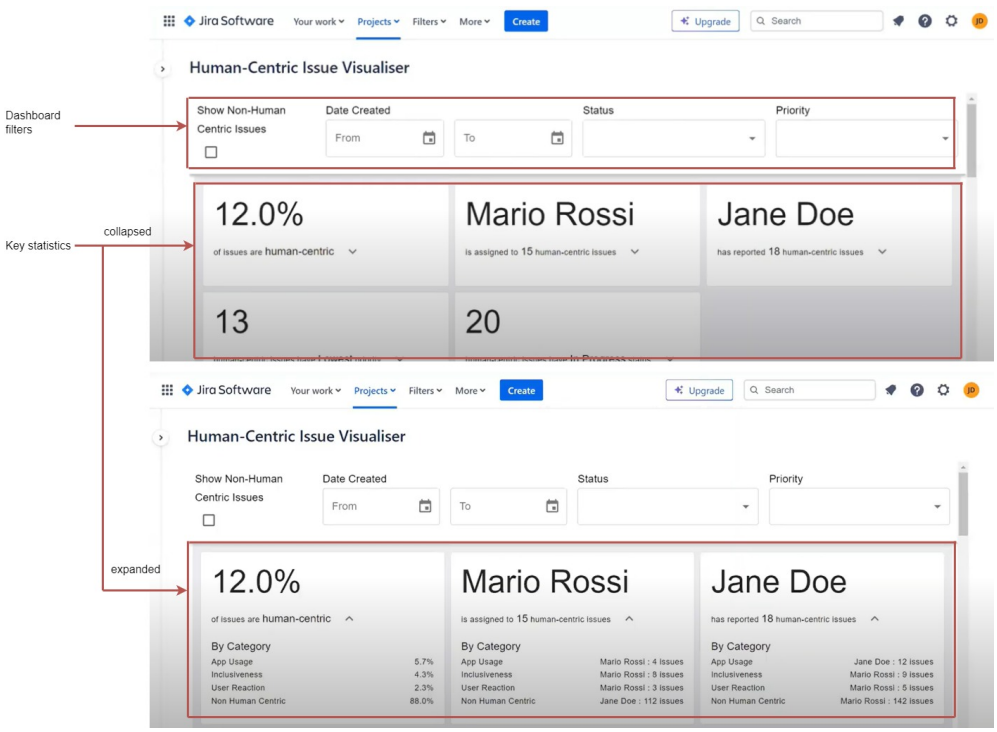

Fig. 13 The filter allows the developer to filter the dashboard to show/hide non-human centric issues and filter by date created, by status, and by priority. The key statistics section of the dashboard shows the statistics. The key statistics section can also be expanded to provide the issues and the counts/percentages according to the HCI category.

prototype's separate React-based interface, these all use built-in Jira chart visualisation support and look and feel like other Jira-based charts. They also allow developers to filter and display charts in a more integrated, natural fashion within Jira.

Developer views HCIs grouped by priority and grouped by status

Fig. 15 shows the "group by" feature of the Jira plugin. This feature allows the developer to view the HCIs grouped according to priority and status. Both views have stacked bar-chart and grouped bar-chart modes. The developer can switch between the modes to see the grouping according to their preference. These use the same concepts of presenting groupings of issues by human-centric issue type as the GitHub prototype. However, as above, these use Jira charting features and provide a much wider array of interactive chart manipulation support, all within the Jira issue tracking environment. They also allow developers to select items in the chart and interactively move to a list of all issues matching the criteria.

Fig. 14 The developer can view the HCI category distribution by overall issues, summary, and descriptions. The view also allows the developer to switch between the bar-chart and pie-chart modes.

Developer views HCI list and their HCI categories

Fig. 16 shows the issue list and their HCI category feature(s). The plugin allows the developer to view the keys of the issues, their summaries, and their HCI categories in a listed view. Unlike the GitHub prototype this uses native Jira issue display and interaction features, providing a more integrated and familiar issue interaction approach.

Developer views HCI details

Fig. 17 shows the issue panel where each section of the issue is labelled by its HCI category with the use of the classifier. The issue panel allows the developer to collapse and expand the issue content. Fig. 17 specifically shows an instance where the comments section of a particular issue is expanded, and its HCI category is shown against it.

5 Evaluation of Prototypes

5.1 GitHub Prototype Evaluation

To evaluate HCIV both qualitatively and quantitatively, we conducted virtual one-on-one moderated testing sessions with volunteers with IT industry experience and

Fig. 15 The developer can switch between the stacked bar-chart and grouped bar-chart modes to see the grouping of the HCIs by priority and the status.

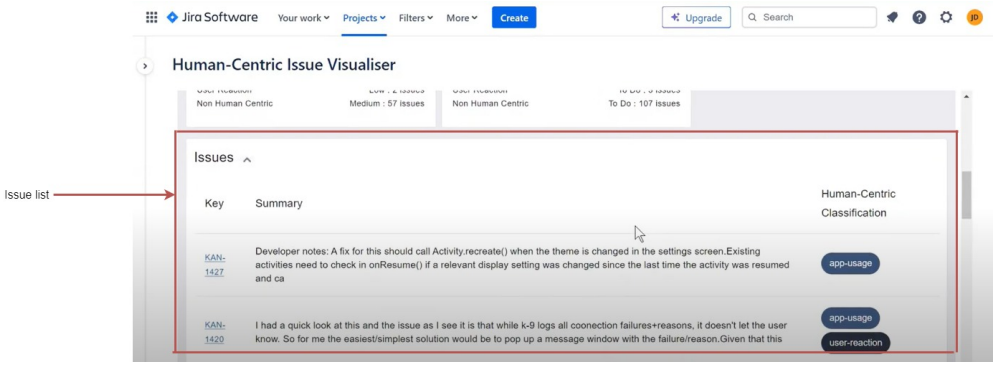

Fig. 16 The developer can view the issue list and their HCI categories.

familiarity with GitHub. Prior to conducting the user study, we obtained ethics approval from the Monash Human Research Ethics Committee (Approval number: 35633).

5.1.1 Evaluation Approach

Participants were recruited from the authors' professional networks via advertising the study on Twitter and LinkedIn, as well as through email and word of mouth. The

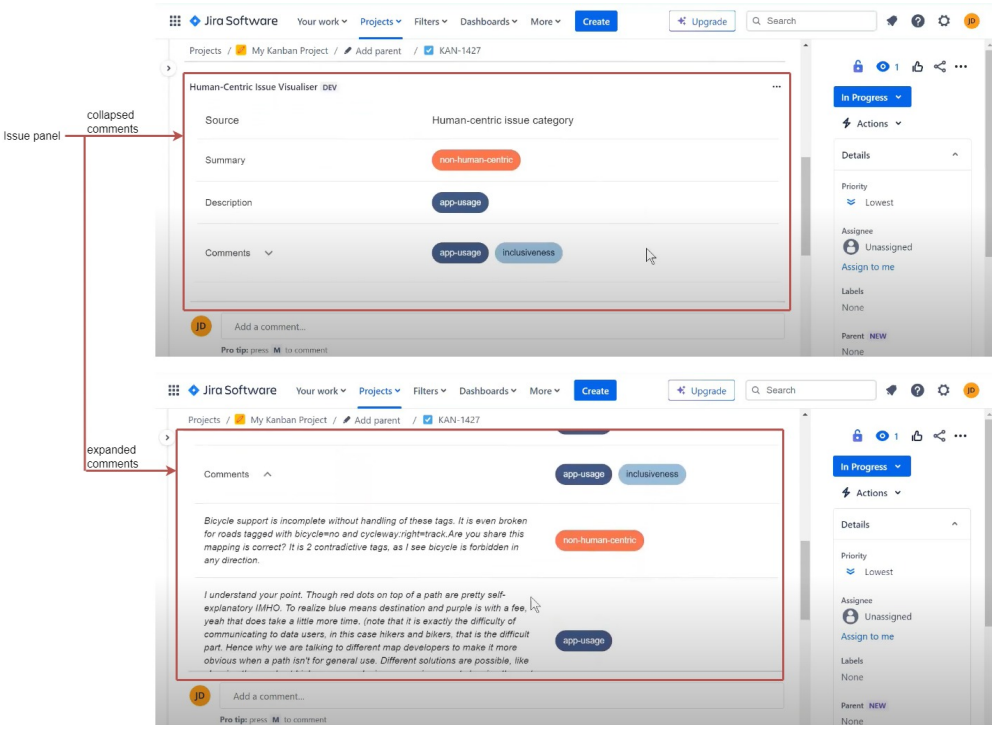

Fig. 17 An instance where the comments section of a particular issue is expanded and its HCI category is shown against it.

user testing sessions were conducted as one-on-one Zoom calls in which the researcher (first author) gave each participant a brief presentation outlining the aim which HCIV tries to achieve - to support software practitioners in human-centric software development. Following this, the researcher gave the participant a short overview of the features of HCIV before providing them with access to a dummy GitHub repository, and HCIV, to use it themselves. The dummy repositories were clones of large open source repositories including bitcoin-wallet/bitcoin-wallet, mozilla-mobile/fenix and signalapp/Signal-Android, each with a small sample of the actual repository's issues copied into the clone, in attempts to replicate a realistic use-case. The participant was encouraged to spend some time familiarising themself with the content of the repository, and the nature of the issues which have been raised in it.

Each participant received a document listing a several tasks that they needed to undertake while using HCIV. This included ask for a repository's issues to be classified, reclassify an issue, re-prioritise an issue, view the issue occurrence history, and view the issue classification summary. They were asked to perform these tasks on a given mock-up repository. The first author was present during the entirety of the participant's opportunity to use HCIV and guide them through various tasks when required. At the conclusion of each session, the participant was provided with access to an anonymous survey on Google Forms, to respond to questions aimed at extracting qualitative and

quantitative feedback about their experience using HCIV, and suggestions for its future direction. We have provided the survey questions and analysed responses online [1].

5.1.2 Participants

Seven participants took part in our user evaluation. Within this group, there was a wide variety of experience in the IT industry. Two participants had more than 10 years of experience, while two had 5-10 years, another two had 2-4 years, and one participant had 0-1 years of experience. Six participants had experience as software developers. Two participants had experience as software testers, and two had been software architects. One participant each had experience in the roles of project manager, QA engineer, scrum master, UX designer and researcher.

5.1.3 Evaluation Results

Of these seven participants, three of them considered HCIs to be a significant point of focus while developing software, two considered them to a very minor extent, and the remaining two participants said that they consider them to a moderate to great extent. The participants were asked what impacts they think that actively considering and attending to HCIs during the software development life cycle may have on software solutions. This question was asked in order to gauge whether the intention behind HCIV was likely to have a meaningful impact on software development, regardless of whether HCIV itself was able to achieve its intention. One participant pointed out that non-technical issues may be identified by considering and attending to HCIs. Another remarked that this practice may reduce the number of issues being found after the launch of the software, which would help to avoid budget and deadline overruns. It was also expressed that the lack of consideration for HCIs during the software development life cycle may result in software, which cannot actually be used as intended in the contexts where it will ultimately be used. This participant mentioned that this can result in expensive change requests, causing revenue loss for the client and productivity losses for the end users.

The following questions were specific to HCIV and intended to quantitatively assess the usefulness of its various features using a rating scale of 1 to 5, where 1 meant that the tool was not useful at all, and 5 meant that the feature was very useful. The best-received feature was the categorisation-correction tool which received five '5' ratings. The worst-received feature was the issue progress tool, which accumulated two '2' and three '3' ratings. The data visualisations on the overview view had a relatively uniform distribution of ratings across the scale, and the remaining features had left (towards '5')-skewed distributions.

Our participants were also prompted to report any difficulties they experienced while using HCIV, yielding a number of usability issues. The following key issues were raised: it was intuitive to try and drag-and-drop the issues on the progress and prioritise views, however, this was not possible, the polar area chart on the overview was not easily understandable, the filter tool on the list view should be more visible,

[1] https://github.com/kashumi-m/ReplicationPackageHCIVUserEvaluation

it was difficult to close the information pop-up, and the categorisation-correction tool had a double-up of text which made it confusing to use.

Following this, participants were asked if they had any suggestions for modifications to existing features or ideas for new ones. One participant made the detailed suggestion to merge the prioritise, categorisation-correction, and list tools, such that the user may change the priority and categorisation of issues on the list view where the full issue body is visible. Another suggestion was to automatically highlight the keywords of each issue body or comment which caused the machine-learning classifier to classify it as human-centric (if it was indeed human-centric). It was also put forth that the non-HCIs may be either not counted by default, or given less weighting in the data visualisations as the focus of HCIV is on HCIs. A suggestion was made to implement a suggested-fixes feature, detailing to users how to solve various common HCIs. One proposal made by multiple participants was to focus on integrating HCIV with an issue and project tracking tool like Jira or Trello which already has fully fledged progress tracking and task assignment features which HCIV currently emulates with less extensive functionality.

When asked if the participants could see themselves realistically using HCIV in an industry setting, two participants said 'No' whereas five said 'Yes'. A more detailed follow-up question tried to gauge if they answered 'Yes', at which stage(s) of the software development life cycle would they be likely to use HCIV. The stages included Planning, Analysis, Design, Implementation, Testing and integration, and Maintenance. Testing and integration was the most popular answer, selected by five participants. Analysis was selected by two, and all other stages were selected by three participants each. For the participants who said that they could not see themselves using HCIV in an industry setting, their reasons were: tools such as Jira and Trello already provided similar and more extensive issue-tracking features, and another participant currently only develops software alone or in small teams, and did not see the need for such a tool in these settings. A more fine-grained question was asked from the participants to see which specific features of HCIV they would not use. The feature most commonly identified as redundant was the reset repository tool, selected by four participants. The progress tool was selected twice, and the data visualisations once.

Overall, participants liked HCIV's ability to link to GitHub repositories, and that they could override the machine-learning categorisation of issues. The capabilities to prioritise and filter through issues were also well received. However, our user evaluation identified a number of key future extensions and directions. A number of user experience improvements are needed to existing features as suggested by the participants, such as integration of HCIV with an issue and project tracking tools such as Jira or Trello. This motivated us to create the Jira prototype as well, which is presented next.

5.2 Jira Prototype Evaluation

We conducted a user study to evaluate the Jira plugin by software professionals. We gathered their feedback qualitatively and quantitatively. Prior to conducting the user study, we obtained ethics approval from the Monash Human Research Ethics Committee (Approval number: 35633).

5.2.1 Evaluation Approach

We used a questionnaire (23 open and closed-ended questions) hosted on Google Forms with a linked video demo of the Jira plugin, and annotated screens with comprehensive details. The participants were asked to provide their feedback based on the material we provided. The questionnaire is available online[2].

We recruited 40 software professionals worldwide using the recruitment platform Prolific. Each participant was paid 4.50 Sterling Pounds for their time. The participants spent 20-30 minutes evaluating the HCIV Jira plugin.

5.2.2 Participants

The details of the participants are given in Table 1. The majority of the participants had 2-4 years of experience (50%; n=20) and were developers (85%; n=34). In order to understand the participants' context more with respect to HCIs (Table 2), we asked them if they consider HCIs during development. On a Likert scale of 1-5, where 1 denoted not at all and 5 denoted it's a significant focus, the majority rated it as 4 (35%; n=14).

The participants shared their opinions (presented below) on the impact of actively considering and attending to HCIs during SDLC on efficient software delivery that satisfies end users.

Efficiency and impact (20%; n=8). Actively considering and attending to HCIs during SDLC has a "high impact on software development life cycle and the efficient delivery" [JP2] according to our participants' opinions. "It will give the software purpose of why it is developed, and we can always make sure that every end user requirement is catered for from start to finish" [JP19] and "It can lead to a faster development cycle and lower the cost. It reduces the rework needed and helps with prioritizing the work, leading to better UX and software that meets the users' needs better" [JP17].

Enhanced user experience, end user satisfaction and usage (20%; n=8). Participants believed that "by prioritizing user needs, we can improve customer satisfaction, gain a competitive advantage, and lower long-term support costs by enhancing user experience, reducing rework, and increasing adoption rates" [JP36]. "It helps (developers) spend less time with any task they are working on, again mostly it satisfies users and ensures that the software it meets expectations" [JP20].

Usability and accessibility (15%; n=6). The participants shared that by actively considering and attending to HCIs during SDLC, "potential usability issues can be avoided" [JP8]. "It allows software to be usable by a wide variety of users with varying skill sets" [JP24], as by "pay(ing) attention to things like how easy it is to use the software, how it looks, and if it works well for everyone, means that the software is more likely to be useful and liked by the people who use it" [JP40].

User-centric design (10%; n=4). "Besides planning and testing, HCI is the most important factor to be considered as it can make or break a product" [JP12]. As the participants shared, actively considering and attending to HCIs during SDLC has a "huge impact. The application is intended for the user, and the user is the ultimate

[2] https://github.com/kashumi-m/ReplicationPackageHCIVUserEvaluation

metric for how well and effectively your application is. To put an example of a feature, it is very important to consider how the user interacts with an application, down to even the minute gestures. This allows us as developers to create a solution that will be widely received and also mean our process will be much more efficient" [JP3].

Integration with SDLC (7.5%; n=3). "There can be a lot of software produced based on attending HCI during the life cycle; after all, the design process is for humans and by humans. There's always feedback on a given project, and more than one time, the feedback may be directed to HCI, so having that in mind during the SW development life cycle could be useful" [JP25]. As "understanding the SDLC is human centered" [JP27], "this would be a feature that we can incorporate into SDLC in the testing and maintaining period" [JP5].

5.2.3 Evaluation Results

Usefulness of the features and the tool. We asked for feedback on the usefulness of the seven features of the plugin and the overall ease/straightforwardness of using the HCIV in SDLC. The results are available in Table 3. The participants rated the features and overall ease/straightforwardness using a Likert scale of 1-5, where 1 indicated not at all and 5 indicated very useful. Except for key statistics, the majority of the participants rated 5 for the features. For key statistics and overall ease/straightforwardness, the majority rated 4. The mean scores for key statistics, issue panel and priority-grouped visualisations were less than for but more than 3.90. However, all had medians equal to or more than 4, indicating the participants' perseverance towards HCIV being useful. The highest rated feature was the *issue list*.

We asked the participants who rated 1-2 for the overall ease/straightforwardness of using the HCIV for the reason for their rating. 3 participants rated 2 out 5 and none rated 1. The shared feedback suggested that the participants found that the presentation and the usability of the tool could be improved, "So the main issue I have with the HCIV is the presentation of it. The top dashboard filters that give the written feedback is nice and I engaged with it effortlessly. The rest of the filters are useful and needed but their presentation could be much better. I think the descriptions of the charts could do with more natural language description" [JP3], there has to be a balance between functionality and clutter" "Having filters would be a great addition to find the relevant issues. But having too many visualizations would clutter the interface and it would be difficult for me to find and prioritize tasks" [JP5], and complexity might lead to a learning curve: "It's moderately difficult to use mainly because of the number or complexities of multiple process within the system but with time, dedication and attention one can easily understand it" [JP32].

Redundant Features. It is important for any software company to have a good understanding of their product on which features are needed and which are not. We collected this feedback from the participants. As given in Table 4, most participants said there are no redundant features or features they won't use (60%; n=24).

Potential Use in Industry Settings. To understand the potential use of the plugin in industry settings, we asked the participants if they could see themselves using it. The majority said yes (90%; n=36). We also asked in which stages they would

Participant ID	SD experience	Role
JP1	2-4 years	Project manager, Software tester
JP2	2-4 years	Developer, QA engineer, Software tester
JP3	2-4 years	Developer, Systems analyst, QA engineer
JP4	2-4 years	Developer, Systems analyst, Project manager
JP5	2-4 years	Developer
JP6	2-4 years	Project manager
JP7	>10 years	Developer
JP8	0-1 years	Developer
JP9	2-4 years	Developer
JP10	5-10 years	Developer, Systems analyst
JP11	2-4 years	Developer
JP12	5-10 years	Developer, Project manager, Software tester
JP13	5-10 years	Developer
JP14	2-4 years	Developer, DevOps engineer
JP15	5-10 years	DevOps engineer
JP16	2-4 years	Developer, Software tester
JP17	5-10 years	Developer, Systems analyst, Software tester
JP18	5-10 years	Developer
JP19	2-4 years	Developer, Systems analyst
JP20	2-4 years	Developer
JP21	2-4 years	Developer
JP22	5-10 years	Developer, Software tester
JP23	0-1 years	Developer, Software tester
JP24	>10 years	Developer
JP25	2-4 years	Developer, Systems analyst
JP26	2-4 years	Developer
JP27	5-10 years	QA engineer
JP28	0-1 years	DevOps engineer
JP29	2-4 years	Developer
JP30	>10 years	Developer, Software tester
JP31	>10 years	Developer, Project manager, Product owner, DevOps engineer
JP32	2-4 years	Developer, Software tester
JP33	5-10 years	Developer
JP34	5-10 years	Developer
JP35	5-10 years	Developer, Project manager, QA engineer, Software tester
JP36	2-4 years	Developer
JP37	5-10 years	Developer
JP38	2-4 years	Developer
JP39	0-1 years	Developer, QA engineer, Software tester
JP40	2-4 years	Systems analyst
Summary of participants' experience		
0-1 years	4	10.00%
2-4 years	20	50.00%
5-10 years	12	30.00%
>10 years	4	10.00%
Developer	34	85.00%
Systems analyst	7	17.50%
Project manager	6	15.00%
Product owner	1	2.50%
DevOps engineer	4	10.00%
QA engineer	5	12.50%
Software tester	11	27.50%

Table 1 Participants' Details. SD: Software development. The majority of the participants had 2-4 years of experience (50%; n=20) and were developers (85%; n=34)

	1	2	3	4	5
To what extent do you consider HCIs when developing software?	4	5	7	14	10
	10.00%	12.50%	17.50%	35.00%	25.00%

Table 2 Participants' consideration of HCIs during development (1=not at all; 5=it's a significant focus). The majority rated it as 4 (35%; n=14).

	1	2	3	4	5	Mean	Median
Dashboard filters	0	3	5	15	17	4.15	4
Key statistics	0	3	5	23	9	3.95	4
HCI category visualisations	0	1	7	14	18	4.23	4
Issue list	0	2	6	12	20	4.25	4.5
Issue panel	0	4	11	10	15	3.90	4
Priority-grouped visualisation	0	4	11	9	16	3.93	4
Status-grouped visualisation	0	2	10	13	15	4.03	4
Overall ease/straightforwardness of using the HCIV	0	3	4	18	15	4.13	4

Table 3 Usefulness of the features and the entire tool. The results indicate that participants believe HCIV is useful.

Are there any **redundant features/features that you would not use**, that the HCIV has?		
Dashboard filters	1	2.50%
Key statistics	2	5.00%
HCI category visualisations	4	10.00%
Issue list	2	5.00%
Issue panel	4	10.00%
Priority-grouped visualisation	4	10.00%
Status-grouped visualisation	5	12.50%
None	24	60.00%

Table 4 Redundant features. The majority of the participants said there are no redundant features or features that they won't use (60%; n=24).

use HCIV. The majority said, in testing and integration (60%; n=24). All results are given in Table 5.

4 participants said that they would not use HCIV in industry settings. The reasons are miscellaneous. [JP3] said they don't have any tools in the pipeline to address human-centric design. Therefore, there won't be any initiative to use HCIV either. [JP7] shared, "missing overview of all issues, highly specific to software involving these type of issues". [JP24] shared their opinion on overcomplicating the SDLC, "While it does seem useful for software that focuses on HCI, the software development cycle I've worked with would not benefit from this, or it would introduce an overcomplicated process for identifying HCIs". [JP36] shared, "We have a lot of tools that display information in JIRA, that it often risks information overload, when really the important information is almost always just the title, description and comments of any particular issue. I can see the HCIV potentially being useful in very large projects, especially those where multiple people control the prioritisation of issues".

	Yes	No
Could you see yourself realistically **using the HCIV in an industry setting?**	36	4
	90.00%	10.00%
If you answered 'Yes', which **stages of the software development life cycle** would you be likely to use it?		
Planning	20	50.00%
Analysis	20	50.00%
Design	18	45.00%
Implementation	19	47.50%
Testing and integration	24	60.00%
Maintenance	17	42.50%

Table 5 Potential Use of HCIV in Industry Settings. The majority said they will use it (90%; n=36) and the majority said they will use it in the testing and integration stage (60%; n=24).

On a scale from 0-10, how likely are you to **recommend** HCIV to a friend or colleague?										
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	0	2	1	1	2	1	7	10	5	11
0%	0%	5%	3%	3%	5%	3%	18%	25%	13%	28%

Table 6 Promoting to a friend or colleague (0=not at all likely; 10=extremely likely). The majority rated 7 or more than 7 (82.5%; n=33).

Net Promoter Score. We asked the participants if they would recommend HCIV to a friend or colleague, and they rated from 0 to 10 (0=not at all likely; 10=extremely likely). The majority rated 7 or more than 7 (82.5%; n=33).

5.3 Threats to Validity

As the user evaluation was carried out with only seven participants, the results obtained are preliminary. There also exists a potential selection bias among the participants, as they were recruited from the researcher's networks and thus may be more likely to have more prior knowledge about human-centric software development from an academic perspective compared to the average software practitioner. This selection bias may have affected the generalisability of HCIV. Users may query about how HCIV would perform in a more diverse real-world environment with more heterogeneous teams. Potential ethical issues around data privacy may exist if sending sensitive issue content to our classifier, though this classifier uses a local BERT classifier and does not store data.

Furthermore, the dummy repositories used in the user evaluations were all clones of large open source projects, with a small subset of their original issues due to rate limiting issues associated with the current prototype. Thus, these repositories are not representative of a wide range of 'real-world' projects including them lacking the scale, complexity, and diversity of real-world projects. Compounding this problem is the fact that the participants, while encouraged to familiarise themselves with the dummy repositories, were almost certainly and understandably not as invested in, or familiar with the repositories to the same extent that a software practitioner would be to the repository of a 'real-world' project they are working on. Moreover, many of HCIV's features, such as the data visualisations and progress tracking tools, are designed to make a meaningful impact on a project over an extended period of time.

Due to time constraints, the user evaluations were conducted as one-off sessions in which the participants used HCIV with a dummy repository, diminishing their ability to evaluate its effectiveness over a long period of time. In addition to that, in the user study, a potential response bias exists as initial the description of HCIV argues for the need to address HCI which follows the question about the impact of considering HCIs in SDLC. The answers the participants have given might have got influenced by the mentioned description.

A limitation of our current prototype is the rate limiting of the GitHub API, which occurs when too many requests are made to a repository in a given period of time. The more issues a repository has, the more requests must be made in order to link HCIV to it. As a result, for repositories with thousands of issues, the prototype will take considerable time to classify and tag all of the issues in the repository. Mitigations for this issue are (i) classify issues in smaller batches e.g. small number of hundreds of issues over a period of hours or days; (ii) classify smaller subsets of issues on a need-to-classify basis; or (iii) start using the tool from project inception.

Unlike the GitHub-integrated version, the Jira-integrated version followed a complete questionnaire-based approach where a video of the tool in action was embedded in the questionnaire for the participants to watch. The participants were expected to answer the questions based on what they grasped from the video. This is different from hands-on experience of using the tool. Therefore, the feedback we received could be different from the feedback the participants might provide if they used the tool for real. In addition to that, if the tool was released as a beta version for a prolonged period, the experience of the potential users could have been captured in terms of feedback and could have been looped in for continuous improvement. Potential ethical issues around data privacy may exist if sending sensitive issue content to our classifier, though this classifier uses a local BERT classifier and does not store data itself.

6 Discussion

To the best of our knowledge, there exists no tool that classifies issues in GitHub or Jira projects according to their human-centric issue categories. The high-level requirements recommended for a human values dashboard by Nurwidyantoro et al. [22] are the closest to HCIV, which we used to map the high-level requirements of HCIV as mentioned at the beginning of this paper. In both user evaluation studies, the majority of the participants mentioned that they would use HCIV in industry contexts/real-world. This implies the potential commercialization of HCIV and also the importance/value of having such a tool to put into practice. Overall, the feedback for the features of both versions of HCIV was positive. We have given recommendations for using HCIV in the next subsection of the paper.

6.1 Benefits of using HCIV

1. **HCIV tags human-centric issues.** Due to its capability of tagging HCIs, HCIV is highly beneficial for both developers and managers to easily identify which are HCIs and which are not. This reduces the time spent on manually tagging each issue in an issue list, which improves the productivity of the developer/team.

2. **HCIV allows developers to group, filter, and view HCIs.** Given the features of grouping, filtering, and viewing HCIs, HCIV gives flexibility for the manager as well as for the developers to get an overview of the existing HCIs for them to address.
3. **HCIV introduces the support for developers to better prioritise and assign HCIs.** Given the view by priority feature, the manager and developers can determine whether the HCIs are better prioritised or need to be changed. This further allows the manager or the team to assign the HCIs to the relevant developers to work on.
4. **HCIV uses issue tracking comprehensively.** HCIV is not only limited to issue tracking but also to bug reporting, handling feature requests, hosting discussions, and processing support requests.
5. **HCIV identifies key HCI spikes during feedback to better identify reasons and address them accordingly.** As feedback is provided by the users/client, HCIV allows the manager and the developer to identify the existence of HCI spikes due to the capability of getting the count. This function helps identify the root causes and address them accordingly so that such spikes are avoided in the future.
6. **HCIV helps better focus the team's efforts to address different criticality of HCIs.** As an issue tracker is used by an entire team, an HCIV integrated issue tracker brings clarity to the team on which types of HCIs exist in the software they develop. This helps the team to focus better and address different critical aspects of HCIs.

6.2 Future research

Misclassification could undermine trust of HCIV. We reused our human-centric issue classifier from our previous work [6], which carried out a large scale evaluation of the classifier's performance on several repositories and a large number of manually tagged issues. This showed the classifier performs very well in classifying human-centric issues accurately across several different size and domain repositories. However, the classifier must be retrained on new repositories to continue to maintain its performance. As users of HCIV can override to correct our classifier's mis-classifications, this provides a promising future work opportunity to retrain and improve our classifier. We aim to further improve the machine-learning classifier's accuracy by utilising HCIV's categorisation-correction tool as an ever-growing source of manually checked training data.

In future iterations of HCIV, we will use an enhanced machine-learning classifier, as the one used in the current version was chosen for its ease-of-use and ease-of-integration, and exists from our previous work. The rate limiting of the GitHub API can be mitigated by retrieving issues from a large project repository in batches and classifying and tagging the repository issues over a period of time. Integration of HCIV with other issue and project tracking tools, such as Trello (in addition to Jira), as one of the other future directions of this research, would be of prime importance to increase the viability of adopting it in an industry setting. Future development and further evaluation of HCIV by releasing beta versions for real-world use will help

researchers improve their understanding of how the awareness and ability to manage HCIs affect the efficient delivery of software solutions that meet the needs of their end users. Similarly, we hope that software practitioners will benefit from the future development of HCIV so that they can be aided in the efficient delivery of software solutions that meet the needs of their end users.

7 Summary

We have presented HCIV, a novel proof-of-concept GitHub-integrated and Jira-integrated issue dashboards that aims to support software practitioners to consider and manage HCIs throughout the software development life cycle. We utilised a machine-learning classifier to identify and categorise HCIs in a GitHub repository and a Jira project. HCIV provides various visualisations and tools for software practitioners to track and manage these classified issues, including re-classifying, prioritisation and tracking fixing. Our user evaluation identified that HCIV will potentially be helpful for supporting the delivery of more human-centric software solutions.

Acknowledgment

Todd, Madampe, and Grundy are supported by ARC Laureate Fellowship FL190100035.

References

[1] Hartzel, K.: How self-efficacy and gender issues affect software adoption and use. Communications of the ACM **46**(9), 167–171 (2003)

[2] Miller, T., Pedell, S., Lopez-Lorca, A.A., Mendoza, A., Sterling, L., Keirnan, A.: Emotion-led modelling for people-oriented requirements engineering: the case study of emergency systems. Journal of Systems and Software **105**, 54–71 (2015)

[3] Stock, S.E., Davies, D.K., Wehmeyer, M.L., Palmer, S.B.: Evaluation of cognitively accessible software to increase independent access to cellphone technology for people with intellectual disability. Journal of Intellectual Disability Research **52**(12), 1155–1164 (2008)

[4] Wirtz, S., Jakobs, E.-M., Ziefle, M.: Age-specific usability issues of software interfaces. In: Proceedings of the IEA, vol. 17 (2009)

[5] Grundy, J., Khalajzadeh, H., Mcintosh, J.: Towards human-centric model-driven software engineering. In: ENASE, pp. 229–238 (2020)

[6] Khalajzadeh, H., Shahin, M., Obie, H.O., Agrawal, P., Grundy, J.: Supporting developers in addressing human-centric issues in mobile apps. IEEE Transactions on Software Engineering **49**(4), 2149–2168 (2022)

[7] Kulyk, O., Kosara, R., Urquiza, J., Wassink, I.: Human-centered aspects. In: Human-Centered Visualization Environments: GI-Dagstuhl Research Seminar, Dagstuhl Castle, Germany, Revised Lectures, pp. 13–75 (2007). Springer

[8] Mathews, C., Ye, K., Grozdanovski, J., Marinelli, M., Zhong, K., Khalajzadeh, H., Obie, H., Grundy, J.: Ah-cid: A tool to automatically detect human-centric issues in app (2021)

[9] Khalajzadeh, H., Shahin, M., Obie, H.O., Grundy, J.: How are diverse end-user human-centric issues discussed on github? In: Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society, pp. 79–89 (2022)

[10] Grundy, J., Khalajzadeh, H., Kanij, T., Mueller, I.: Humanise: Approaches to achieve more human-centric software engineering. In: Evaluation of Novel Approaches to Software Engineering: 15th International Conference, ENASE 2020, Prague, Czech Republic, Revised Selected Papers, p. 444 (2021). Springer Nature

[11] Ortu, M., Murgia, A., Destefanis, G., Tourani, P., Tonelli, R., Marchesi, M., Adams, B.: The emotional side of software developers in jira. In: Proceedings of the 13th International Conference on Mining Software Repositories. MSR '16, pp. 480–483. Association for Computing Machinery, New York, NY, USA (2016)

[12] Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D.M., Damian, D.: The promises and perils of mining github. In: Proceedings of the 11th Working Conference on Mining Software Repositories, pp. 92–101 (2014)

[13] Ko, A.J., Chilana, P.K.: Design, discussion, and dissent in open bug reports. In: Proceedings of the 2011 iConference, pp. 106–113 (2011)

[14] Twidale, M.B., Nichols, D.M.: Exploring usability discussions in open source development. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, pp. 198–198 (2005). IEEE

[15] Yusop, N.S.M., Grundy, J., Vasa, R.: Reporting usability defects: A systematic literature review. IEEE Transactions on Software Engineering **43**(9), 848–867 (2016)

[16] Andreasen, M.S., Nielsen, H.V., Schrøder, S.O., Stage, J.: Usability in open source software development: opinions and practice. Information technology and control **35**(3) (2006)

[17] Pletea, D., Vasilescu, B., Serebrenik, A.: Security and emotion: sentiment analysis of security discussions on github. In: Proceedings of the 11th Working Conference on Mining Software Repositories, pp. 348–351 (2014)

[18] Dabbish, L., Stuart, C., Tsay, J., Herbsleb, J.: Social coding in github: Transparency and collaboration in an open software repository. In: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work. CSCW '12, pp. 1277–1286. Association for Computing Machinery, New York, NY, USA (2012)

[19] Dam, H.K., Savarimuthu, B.T.R., Avery, D., Ghose, A.: Mining software repositories for social norms. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 2, pp. 627–630 (2015)

[20] Novielli, N., Calefato, F., Lanubile, F.: Towards discovering the role of emotions in stack overflow. In: Proceedings of the 6th International Workshop on Social Software Engineering. SSE 2014, pp. 33–36. Association for Computing Machinery, New York, NY, USA (2014)

[21] Cabrera-Diego, L.A., Bessis, N., Korkontzelos, I.: Classifying emotions in stack overflow and jira using a multi-label approach. Knowledge-Based Systems **195**, 105633 (2020)

[22] Nurwidyantoro, A., Shahin, M., Chaudron, M., Hussain, W., Perera, H., Shams, R.A., Whittle, J.: Towards a Human Values Dashboard for Software Development: An Exploratory Study. Association for Computing Machinery, New York, NY, USA (2021)

[23] Barcellini, F., Détienne, F., Burkhardt, J.-M., Sack, W.: A socio-cognitive analysis of online design discussions in an open source software community. Interacting with computers **20**(1), 141–165 (2008)