16th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE2021), online, 26-27 April, 2021

Impact of End User Human Aspects on Software Engineering

John C. Grundy[1][a]

[1]*Department of Software Systems and Cybersecurity, Faculty of Information Technology,*
Monash University, Melbourne, Australia
john.grundy@monash.edu

Keywords: Software Engineering, Stakeholders, End Users, Human Aspects, Human Factors

Abstract: Software is designed and built to help solve human problems. However, much current software fails to take into account the diverse end users of software systems and their differing characteristics and needs eg. age, gender, culture, language, educational level, socio-economic status, physical and mental challenges, etc. I give examples of some of these diverse end user characteristics and the need to better incorporate them into requirements engineering, design, implementation, testing, and defect reporting activities in software engineering. I report on some of our work trying to address some of these issues, including: use of personas to better characterise diverse end user characteristics; extending requirements and design models to capture diverse end user needs; analysis of app reviews and JIRA logs to identify problems and ways developers try to address them; analysis of approaches to improve the accessibility of software designs for diverse end users; adaptive user interfaces and model-driven engineering with human aspects; improved human-centric defect reporting approaches; and use of living lab co-design approaches to ensure end users are first class contributors during all phases of software development. I finish by outlining a research roadmap aiming to improve the incorporation of end user human aspects into software engineering.

1 INTRODUCTION

Humans build software systems to help solve human problems, be they for industry, leisure, health and well being, social interactions, and so on (Rashid et al., 2017; Strengers and Kennedy, 2020; Curumsing et al., 2019; Grundy and Grundy, 2013). Yet, humans are different in many ways, including having diverse age, gender, culture, language, educational level, technical proficiency, preferences in interaction and problem solving styles, personality, emotional reaction to using software, mental and physical challenges, and so on (Burnett et al., 2016; Perez, 2019; Grundy, 2020; Curumsing et al., 2019; Cruz et al., 2015).

There has been considerable research into cooperative and human aspects of software engineering – from the perspectives of software engineers as humans – for many years (Cruz et al., 2015; Lenberg et al., 2015; Hidellaarachchi et al., 2021). However, there has been much less research into how diverse *END USER* human aspects impact software engineering from requirements, design, implementation, testing, deployment and defect reporting and fixing perspectives (Lopez-Lorca et al., 2014; Alshayban et al., 2020; Grundy, 2020; Hidellaarachchi et al., 2021; Burnett et al., 2016; Yusop et al., 2020). While human-computer interaction, design science, psychology, anthropology, and other disciplines have researched these software end user human impacts for many years, particularly on design and usability evaluation, few software engineers know about many of the theories involved, techniques and tools invented, and integrate little or none of these findings into contemporary software engineering practices (Yusop et al., 2016; Grundy, 2020; Madampe et al., 2020; Curumsing et al., 2019; Alshayban et al., 2020; Cruz et al., 2015).

Given the increasing use of software for all aspects of modern living and working, and the increasing diversity of end users and end user challenges this software thus has, software engineers must better understand and take account of diverse end user human aspects. In this keynote talk I discuss how the lack of accounting for some of these end user human aspects produces not-fit-for-purpose software solutions. I then discuss some of the work my HumaniSE (Human-centric Software Engineering) team[1] is doing to address different current deficiencies dur-

[a] https://orcid.org/0000-0003-4928-7076

[1] https://www.monash.edu/it/humanise-lab

ing requirements engineering, design, implementation, evaluation and the overall software process.

The rest of this paper is organised as follows. Section 2 outlines some of the key end user human aspects and issues of failing to address them during software development. I then discuss in Sections 3, 4, 5 and 6 some of our work in improving requirements engineering, design, model-driven engineering and evaluation by incorporating end user human aspects into key steps of development. In Section 7 I discuss progress to date and key planned future work to extend this research and to deliver practical outcomes for software engineers. In finish in Section 8 with key conclusions from this keynote talk.

2 END USER HUMAN ASPECTS

The term *"human factors"* has been used extensively in HCI and Engineering and Design disciplines to characterise human issues involved in designing and using complex technologies (Woodson et al., 1992). The term *"human aspects"* has been used for some time in software engineering but usually to discuss human issues of software engineers, as individuals and teams (Hazzan and Tomayko, 2005). In this paper I use the term *"end user human aspects"* – or just *human aspects* – to describe human issues of the *target end users* of the software systems we build. While this concept is related to human aspects of the software engineers building the systems – and I return to this interplay at the end of this keynote talk – they have quite different implications. Similarly while human factors of complex systems have been well-studied for many years, how to best support *software engineers* in addressing the diverse human aspects of their *software end users* is a related but quite different – and under-researched – focus area.

A wide variety of end user human aspects impact software and its usage. This is becoming more prominent as more and more people need to use software solutions more and more often (Lopez-Lorca et al., 2014; Grundy, 2020). Below I summarise some, but by no means all, of these *end user human aspects* that software engineers increasingly need to consider and address effectively in their work.

Age: End users have a wide range of ages from the very young to the very old. Different aged people have quite different challenges in using software, and may have quite different expectations and reactions to the same software (Grundy et al., 2018; Nouwen et al., 2015; Williams et al., 2013). Failure to take account of differently aged end users may result in software that uses wrong terminology, poor interfaces and workflow, is overly complex or confusing, and is not sufficiently enjoyable or engaging.

Physical and Mental Challenges: Many software solutions have been developed to assist with people living with physical and/or cognitive challenges. Some must be designed and implemented to specifically take account of them to assist with accessibility of the software or to ensure the software supports and not harms vulnerable end users (Carcedo et al., 2016; Sierra and Togores, 2012). These challenges include poor mental health, various degrees of cognitive impairment and a wide variety of physical challenges, such as limited or impaired sight, hearing, mobility and speech (Alshayban et al., 2020; Stock et al., 2008; Zhao et al., 2020; Rashid et al., 2017). Failure to account for diverse physical and mental challenges results in accessibility problems, but also inefficient and ineffective software solutions, confusing and even dangerous systems, end user frustration, and creates digital living barriers and obstacles that add to and/or exacerbate physical ones.

Emotions: Using the same software system often generates positive and negative emotions in different people. For example, positive reactions might include a home monitoring system providing a feeling of safety, to negative reactions to the same software, such as feeling lack of control or being monitored intrusively (Curumsing et al., 2019). These reactions can have a major impact of acceptance and use of solutions, but are very poorly supported by most existing software requirements and design methods (Miller et al., 2015; Taveter et al., 2019).

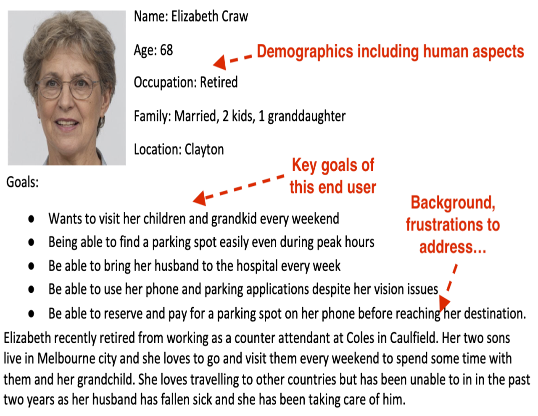
Personality and Cognitive Style: There has been relatively little research about the impact of personality and cognitive style differences of end users on software usage to date (McElroy et al., 2007; Barnett et al., 2015; Burnett et al., 2016). Studies with software engineers and others have shown significant impacts of personality and different cognitive styles – these may thus have a major impact on end user perceptiosn and usage of software solutions (Cruz et al., 2015; Kanij et al., 2015).

Engagement and Entertainment: Many end users are highly driven by enjoyment, entertainment and 'fun' aspects of using software – such as with computer games and gamification-based approaches. Similarly, different people may engage with the same software at different levels and in different ways (Fensel et al., 2017; Kumar, 2013). Failure to take these differences into account will likely result in less appealing software for different end user groups.

Human Values: These include values such as inclusiveness, equality, privacy, openness, etc. (Winter et al., 2018). These human values have been found to be mis-aligned with software they use, with many software systems and developers have values that conflict with one or more end user values (Obie et al., 2021). As a result, these mis-aligned values can cause severe expectation mis-matches of end users and their software solutions, reducing the software take-up and usage.

Gender: Gender bias has been shown to be highly problematic in many modern technologies including many software systems (Perez, 2019; Strengers and Kennedy, 2020; Burnett et al., 2016). Different problem solving styles of different genders have been shown to have a major impact on software acceptance and usability (Burnett et al., 2016).

Ethnicity and Culture: Culture can be used to describe different beliefs and behaviours of different groups of people (Alsanoosy et al., 2019). Software that is biased in terms ethnicity of people is highly problematic, especially for many emerging AI-based smart living and surveillance systems used by police and other agencies (Garvie and Frankle, 2016). Like many other human aspects, software developers often have different ethnicity and culture than many of their target end user groups. Current software development methods do not assist developers in better understanding and accounting for culture and ethnic differences of their end users.


Demographics including human aspects
Key goals of this end user
Background, frustrations to address...
Name: Elizabeth Craw

Age: 68

Occupation: Retired

Family: Married, 2 kids, 1 granddaughter

Location: Clayton

Goals:

- Wants to visit her children and grandkid every weekend
- Being able to find a parking spot easily even during peak hours
- Be able to bring her husband to the hospital every week
- Be able to use her phone and parking applications despite her vision issues
- Be able to reserve and pay for a parking spot on her phone before reaching her destination.

Elizabeth recently retired from working as a counter attendant at Coles in Caulfield. Her two sons live in Melbourne city and she loves to go and visit them every weekend to spend some time with them and her grandchild. She loves travelling to other countries but has been unable to in in the past two years as her husband has fallen sick and she has been taking care of him.

Figure 1: Part of a persona used for smart parking app development

Language: Language differences occur in several forms in end users of software. People speak different languages, use different forms of jargon, have differing educational attainment levels, have different language competencies, and use different dialects and slang (Roturier, 2015). Failure to take these language differences into account means software is much harder to understand and interact with for many end users. Again, software engineers lack sufficient techniques and tools to help them take into account these end user language differences.

Socio-economic status: there is a huge digital and physical divide between those with jobs, money and luxuries, and those in precarious living, little/no work, and who struggle to access some of the necessities of life (Ahmed, 2007). There is still a massive imbalance in the world's wealth and access to physical world as well as digital services, including access to community support services via digital means (Grundy and Grundy, 2013). Most software engineers are relatively wealthy and highly educated – supporting more vulnerable members of society in software solutions is not always straightforward, obvious or even considered (Newman et al., 2015).

3 HUMAN-CENTRIC REQUIREMENTS ENGINEERING

I outline several projects we are undertaking to enhance requirements engineering practices and tools to better support diverse end user human aspects.

3.1 Use of Personas to Model Users

Software engineers need to better understand the wide differences in their end users, in terms of their diverse human aspects outlined in Section 2 as well as others (Lopez-Lorca et al., 2014; Grundy, 2020). One approach used extensively in human-computer interaction and design domains, but less frequently in software engineering, is the persona. Figure 1 shows an example persona from a smart parking app development project one of our student teams and PhD student developed to aid them in better understanding the range of end users of this app. This example shows an elder woman user and outlines her demographics, goals and frustrations relating to transport and parking. We developed several diverse personas and used them to help capture richer requirements, develop a more complete design to support all these possible end users, and to evaluate and refine the prototype new smart parking app. We aim to develop tools to help build richer end user personas with diverse human aspects, and to support their use more extensively throughout phases of software development .

3.2 Identification and Dialogue with Stakeholders

Related to persona building to represent end users, we are working on developing improved approaches to identify software stakeholders in general, not just end users of the software. For example, some e-health sofwtare systems are not used directly by hospital managers or carers of patients, but these are critical stakeholders whose needs from the software also needs to be carefully identified, modelled and taken into account. These stakeholders themselves have diverse human aspects that may different from end users. Improving dialogue with stakeholders and end users to more effectively identify and capture human aspects is needed (McManus, 2004). We plan to capture these with personas and extended modelling languages (see Section 4 below for an example).

3.3 Extraction of Human-centric Requirements from Documents

There has been much work done on extracting functional and non-functional requirements from natural language documents over many years (Osama et al., 2020). We want to explore the identification, extraction, modelling and reasoning about requirements relating to human aspects of end users and related needs. For example, extracting the requirements relating to smart parking for the persona illustrated in Figure 1 semi-automatically would assist software engineers in identifying and using these end user human aspect-related requirements. We also plan design critics that assist software engineers in exploring end user human aspect-related requirements gaps, incomplete requirements or seemingly-erroneous requirements for the domain/target end users (Ali et al., 2013).

4 HUMAN-CENTRIC DESIGN

Following on from the previous section, I discuss some our current projects relating to enhancing the design phase of software engineers to better address end user human aspects.

4.1 Modelling Human-centric Aspects of End Users

Figure 2 shows a wireframe model that one of our student teams has extended to identify different target end user groups and to describe different interfaces needed for these diverse groups (Jim et al., 2021). In this example we model different age group needs from an app interface that needs to take into account age-related differences of these end users. These include different language, font, colour, layout, size

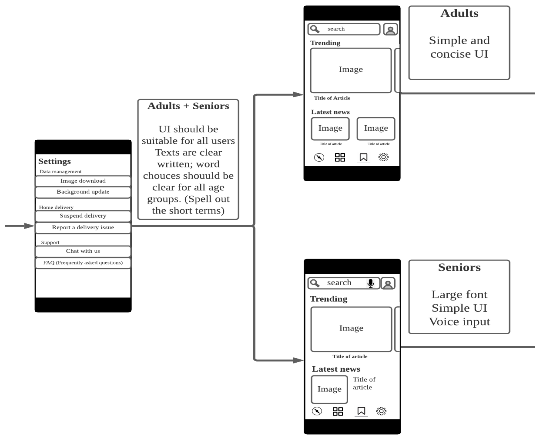


Figure 2: Modelling human-centric characteristics of end users (from (Jim et al., 2021))

and other UI design choices. Such design differences are not just restricted to appearance/interaction design choices but also overall problem solving workflow for (parts of) the app. We want to generalise this approach to other modelling languages for requirements and design – such as i*, UML, BPMN, user stories and so on – to enable better capture and usage of a wide range of diverse end user human aspects during software development.

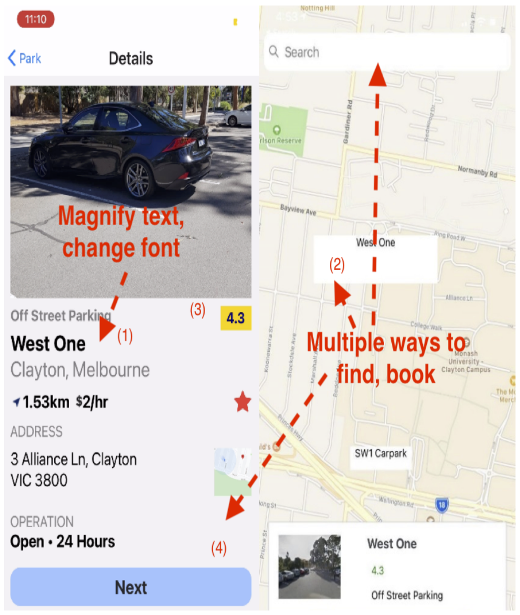
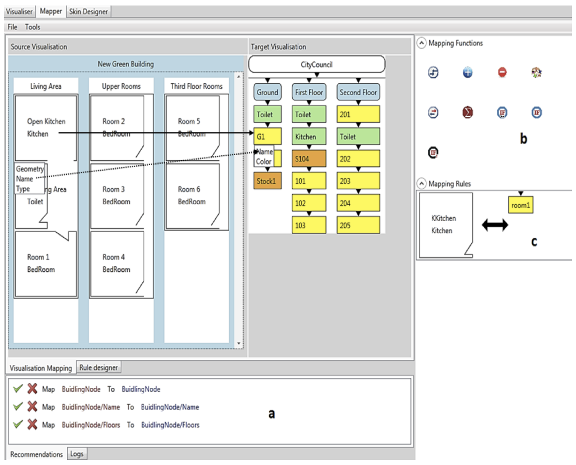


Figure 3: Example of human-centric parking app design

4.2 Design with Personas

In Figure 1 we showed one of our smart parking app personas. Figure 3 shows some of the screens from



✓ ✗ Map BuidlingNode To BuidlingNode
✓ ✗ Map BuidlingNode/Name To BuidlingNode/Name
✓ ✗ Map BuidlingNode/Floors To BuidlingNode/Floors

Figure 4: Example of user-defined data visualisation (from (Avazpour et al., 2015))

the redesigned smart parking app that one our student teams prototyped using a range of diverse personas. Some users have physical and age-related challenges e.g. eye-sight, colour-blindness, mobility skills – that all need to be carefully designed in and evaluated. Others have diverse parking needs relating to human aspects e.g. their children, job, living location, personal preferences, personal emotions and preferences, and language and cultural differences. All of these need to be elicited, modelled and then used during design, implementation and evaluation of the app prototype. In this example, some text is enlarged for users with eyesight challenges (1); different prebooking, selection and driving direction approaches are provided based on different end user needs and preferences (2); image-based recognition of license-plate avoids typographical errors (3); and different reserving/booking/leaving workflows support different user personal circumstances (4).

4.3 Addressing human values and accessibility of web sites and apps

We have several projects looking at analysing web sites and mobile apps for issues with human values violations (Obie et al., 2021), privacy and accessibility issues (Haggag et al., 2021), and user with various human aspect-related challenges e.g. sight, hearing, ageing and so on (Grundy, 2020; Grundy et al., 2018). We are also conducting indepth surveys and interviews of developers and end users exploring reasons for these issues, why they are challenging to address, and how they are fixed when found (Shamsujjoha et al., 2021). We aim to determine from this analysis key design failures in the apps and web sites which can be detected during design-time via improved techniques and tools for developers. We also aim to equip developers with improved guidance on how to correct these issues with improved design decision guidelines and implementation automation tools.

5 HUMAN-CENTRIC MODEL-DRIVEN ENGINEERING

5.1 End-user visualisation development

For many years we have been developing tools to support end-user development of complex software systems (Khalajzadeh et al., 2020; Hirsch et al., 2010; Grundy et al., 1998). This helps to avoid the classic problem with conventional software engineering – reliance on a highly trained expert workforce (software engineers) to build and make necessary changes to software systems. Instead, end users can specify and generate their own solutions, often for very complex domains, incorporating their own human aspect-related needs into their own solutions.

Figure 4 shows an example of one such tool, ConVErT, used by end users to specify and generate complex information visualisations and data mapping systems (Avazpour et al., 2019; Avazpour et al., 2015). This example shows two end user-specified data visualisations – a building layout (left) from a CAD tool dataset and categorisations tree from an ERP systen (middle). A data mapping between parts of each data model can be specified by drag and drop between elements (arrows connecting), used to generate code to implement CAD tool to ERP system data export. Such end user development tools allow end users to tailor their software solutions themselves without waiting for software engineers to do it.

5.2 Adding human aspects to code generators

Figure 5 shows an example of the Visual Care Plan Modelling Language (VCPML) tool in use to specify a care plan for those with obesity (Khambati et al., 2008). This provides a set of visual languages allowing end users – clinicians – to specify complex health care plans. They can then tailor generic care plans to specific patient needs and have the tool generate a fully working mobile app from the specified care plan and associated desired screen definitions. While this idea is good in theory, the VCPML tool

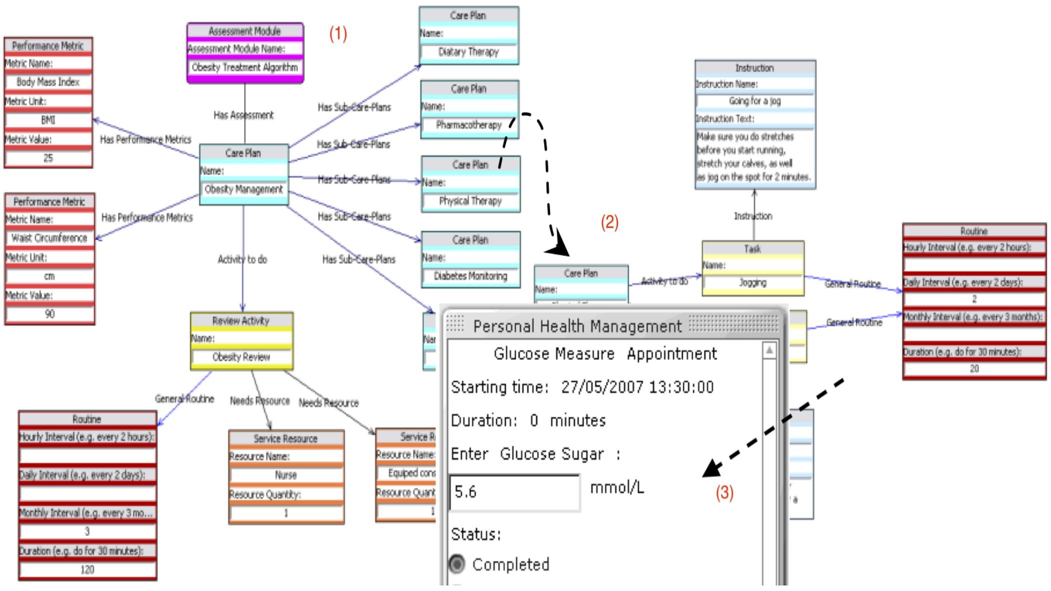


Figure 5: Need for human aspects in MDE (from (Khambati et al., 2008))

does not capture any specific human aspects of different users e.g. different language, gender, age, culture, accessibility issues etc. The tool thus generates a one-size-fits-all mobile app. We are working on incorporating such human aspects of different end users into the modelling languages and code generators of such MDE-based tools. We will then be able to tailor the generated code to range of different individual end user human aspect-related differences, better addressing these in a "personalised" app generated for each different user.

5.3 Adaptive User Interfaces

Related to the above generated app example, we have also been experimenting with using an alternative approach of highly run-time adaptable UI components (Grundy and Hosking, 2002; Grundy and Zou, 2004). These allow different end users to specify a range of preferences e.g. fonts, size, colour scheme, colour mapping, language preferences etc and have their web site or app adapt at run-time to these preferences. Figure 6 shows two example configuration approaches. A large configuration menu (1) allows users to specify a wide range of sight- and cognition-related constraints that need to be satisfied by the web site e.g. colour blindness, eyesight limitations, dyslexia, etc.

Figure 7 shows an example of an adaptive web site prototype using these configurations to adapt a food information website to a particular end user's human aspects e.g. enlarged font size, dyslexia-friendly font face, color blindness friendly colour scheme, colour

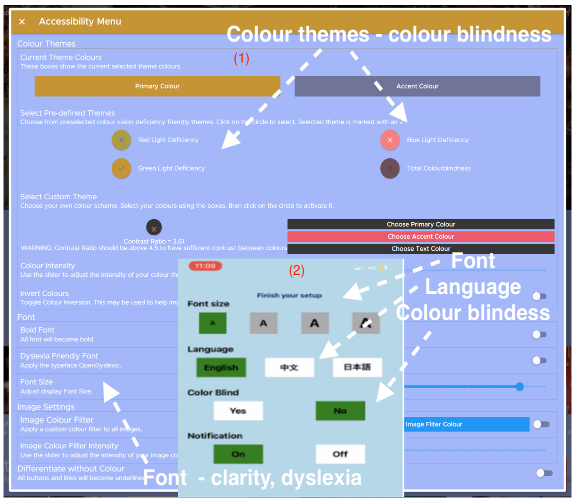


Figure 6: Approaches to UI/app configuration

blindness friendly image colour filtering and site auto re-layout. The second small app configuration panel in Figure 6 is used by the smart parking app prototype from Figure 3 to adapt various UI components in the app to different user human aspect needs e.g. language, font size, style, colour scheme etc. We are also exploring AI-based semi-automated adaptation of UI components to different user human aspect needs, to compliment these manual configuration approaches.

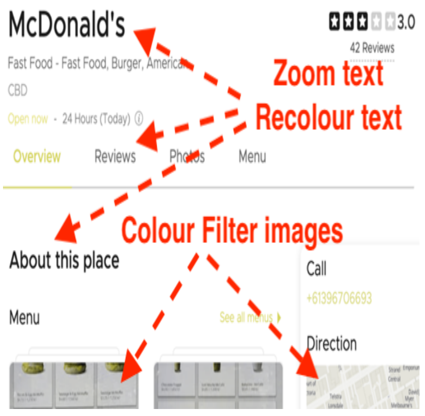


Figure 7: Example of adapted UI

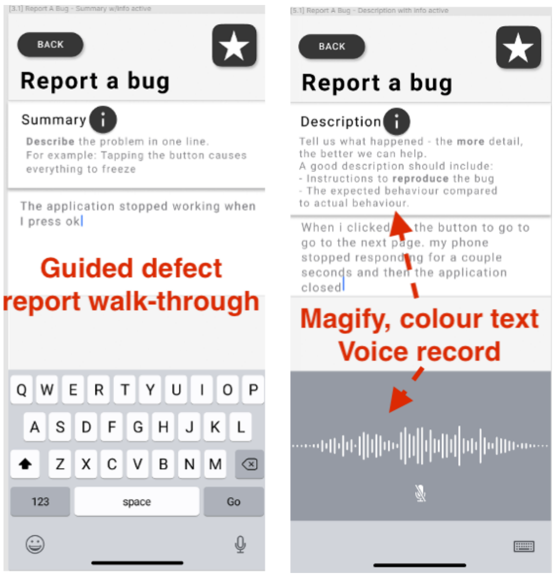
6 HUMAN-CENTRIC SOFTWARE EVALUATION

6.1 A new Human-centric Defect Reporting Taxonomy

We discovered when exploring the domain of usability defect reporting that existing usability defect classification schemes are severely limited (Yusop et al., 2020). To this end we developed a revised usability defect classification taxonomy to aid the development of improved usability defect reports (Yusop et al., 2018). In a similar way, we want to develop a taxonomy of end user human aspects in software engineering (Grundy, 2020), and use these to improve the classification of "end user human aspect defects" in software applications. Such a taxonomy will aid end users in reporting human aspect-related software defects in their applications, but also aid software engineers in understanding these defects, the end users reporting them, what impact they have, and how they might be fixed. This work also leverages our use of personas to describe different end users of software application and analysis of app reviews to determine human aspect and human value defects with apps, both described above.

6.2 Human-centric Defect Reporting

An interesting challenge for end users in reporting human aspect-related defects in their software is the poor usability and human aspect defects in current defect reporting tools! Similar to the poor usability and suitability of current usability defect reporrting tools (Yusop et al., 2018; Yusop et al., 2020), we need to provide end users with human aspect defect reporting tools. Figure 8 shows an example of a human aspect defect reporting prototype tool developed by one


[3.1] Report A Bug - Summary w/info active

Report a bug

Summary

Describe the problem in one line.
For example: Tapping the button causes everything to freeze

The application stopped working when I press ok

[5.1] Report A Bug - Description with info active

Report a bug

Description

Tell us what happened - the **more** detail, the better we can help.
A good description should include:
- Instructions to **reproduce** the bug
- The expected behaviour compared to actual behaviour.

When i clicked the button to go to go to the next page. my phone stopped responding for a couple seconds and then the application closed

Figure 8: Example of human-centric defect reporting

of our student teams. This uses an end user human aspect preference setting panel, similar to the smart parking and adaptive user interface component ones in Figure 6, to configure the reporting tool to different end user sight, hearing, cognitive and language challenges. The left hand side image shows the first stage of reporting a defect. The right hand side image shows further defect information being specified. It also shows an alternative way of capturing this information, using more "human-centric" voice recording and voice transcription. This recorded voice approach can better suit some end users reporting defects than typing a lot of text. We use a set of personas to represent different groups of end users with different human aspect challenges to group these defect reports. When shown to the software engineers, these personas help them to see the defects from the perspectives of end users with different human aspects to each other and to the software engineers themselves. The idea is to help the software engineers reproduce – and fix and test – the defect through the eyes of the diverse defect reporters.

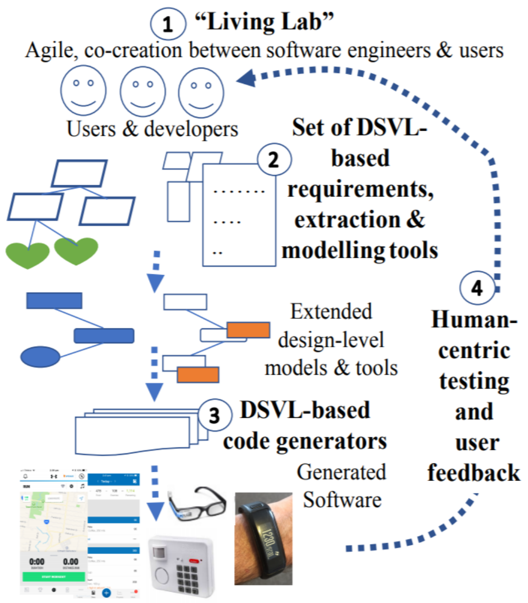
6.3 Developer Surveys and GitHub, JIRA Developer Discussion Analysis

We are interested to better understand (1) what end user human aspects software engineers have experience (or little experience) addressing in their software; (2) which aspects they find harder to address and why; (3) how they currently talk about human aspect-related requirements and defects; and (4) what tools and techniques they currently use to help them address different end user human aspects in their software. To this end, we have been carrying our a large survey and targeted interviews of software engineers. We have also been looking for JIRA and GitHub issue logs and StackOverflow posts that reflect discussion of different human aspects by software engineers. We hope to link these to end user defect reports and app reviews and better understand how these are understood (or not) and addressed (or not) by software engineers. This, we hope, will lead us to better guidelines, techniques and tools for software engineers to improve end user human aspect defect fixing, but also improved design and implementation decision making to avoid or reduce the severity of these human-centric defects.

6.4 Living Lab Co-design Approach

A key issue we have identified with realising our vision of achieving improved addressing of end user human aspects in software engineering is the current approach used to develop software. Despite agile software development's promise of people-oriented development (Hoda et al., 2018), there is still a *"them vs us"* approach taken by software engineers, us = software engineers and them = everyone else. We are exploring the use of a living-lab based approach to software development, popularised by development approaches used for eHealth systems among others (Andersen et al., 2017).

Figure 9 outlines our approach (Grundy et al., 2020). (1) We build focus groups of stakeholders and developers to co-create software solutions as equals. (2) We enrich software requirements and design processes, models, techniques and tools to capture and reason about diverse human aspect-related needs of stakeholders and end users. (3) We use model-driven engineering techniques to generate and configure software allowing for faster development and response to emerging end user needs. (4) We support more human-centric defect reporting and fixing, via proactive feedback loops.



1
"Living Lab"
Agile, co-creation between software engineers & users
Users & developers
2
Set of DSVL-based requirements, extraction & modelling tools
Extended design-level models & tools
3
DSVL-based code generators
Generated Software
4
Human-centric testing and user feedback

Figure 9: Living lab co-design approach (from (Grundy et al., 2020))

7 DISCUSSION

I summarise some of our key findings and progress to date, key gaps still to address, and briefly discuss software engineers and their human aspects.

We have been focusing on better understanding current developer approaches to addressing their end users' human aspects, key open challenge areas in this domain, and then researching new techniques and tools to address these. We conducted a detailed survey, answered by 59 developers and managers, and interviewed 12 developers, to better understand these issues from a software engineering perspective. Figure 10 summarises their rating of key critical end user human aspects in their work. Figure 11 summarises the key reasons given why they find these end user human aspects challenging to address. Key reasons included the different languages and (lack of) comfort with technology of different user groups; range of end user human differences that exist; complexity of user interfaces; different problem solving styles of many end user groups; and differences in terminology used, digital literacy and need to carefully consider text and icon usage.

We asked developers what would help them to improve development of their software to better address some of these diverse end user human aspects. Examples we were given included: the need to develop better development processes to improve target end user involvement in software development; provide developers with better guidelines and practices to follow to address diverse end user human aspects; better requirements capture and human aspect modelling support; AI-based tools to automatically advise on missing end user human aspect issues; more live testing with representative end users; a need for better education of software engineers about diverse end user human aspects and their impact on software usage; simpler interfaces in software for many end user groups; better defect reporting to enable end users to more easily identify, describe and report problems they have with their software; better participant recruitment approaches ensuring more diverse end users

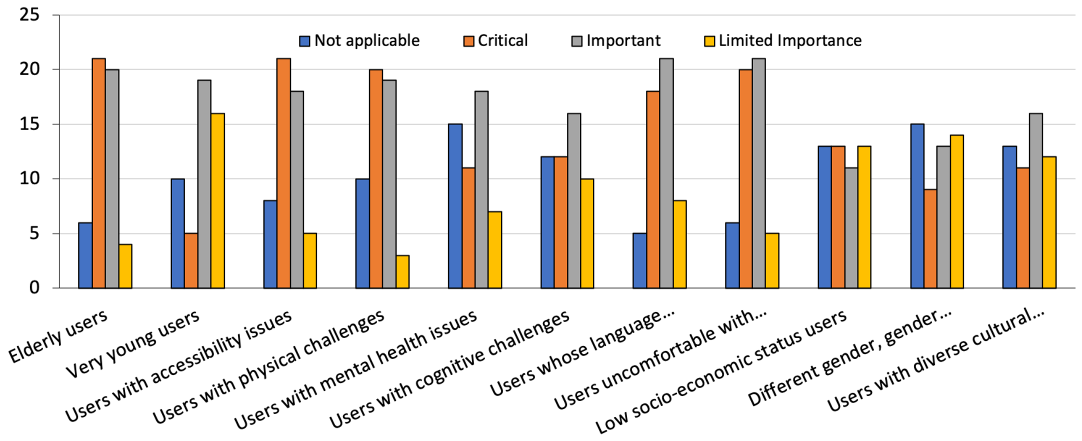


Figure 10: End user human aspects survey respondents judge to be critical (or not) in their work

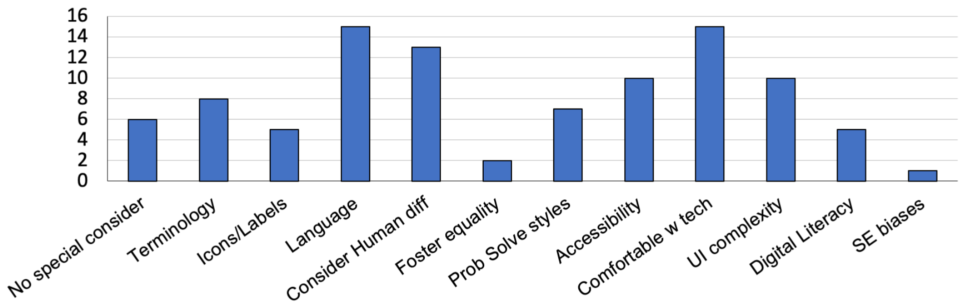


Figure 11: Reasons these human aspects are challenging and how taken into consideration in survey respondents work

are included; better design frameworks and tooling to address a range of end user human aspects; and more research into human aspects in software engineering.

To address some of these issues, we have been developing the techniques and tools outlined in this talk, among many others. Key examples include:

- improved processes to including diverse end user perspectives – living lab based approaches
- better identification of stakeholders and improving capture and modelling of diverse stakeholder and end user human aspect-related requirements
- improved design and implementation support, ranging from better tools to model end user human aspects, include these in software code generators and run-time adaption frameworks, better design guidance for developers to address different end user human aspects in software, and AI-based tools to proactively deploy this guidance
- improved human-centric defect reporting tools, with improved support for developers to locate human aspect-related defects, identify possible fixes, make fixes, and inform end users of fixes made

Some of the key remaining gaps in this area we have identified include:

- lack of a taxonomy of end user human aspects including keywords, phrases and examples – this makes talking about end user human aspects more difficult and impacts all of the techniques and tools we want to provide software developers
- lack of studies focusing on how **software engineers** and software engineering teams address end user human aspects in software – many design and HCI works exist but few if any have made their way into impacting software engineering practice, for various reasons
- lack of tools to identify challenging end user human aspects to address during requirements engineering, including extraction, modelling, 3Cs checking, and validation
- a range of design and evaluation guidelines and tools but lack of connectivity, consistency, and applicability of these tools in many domains e.g. for mobile app development
- overly-complex, inaccessible and incomplete design and implementation guidelines to address many challenging end user human aspects
- difficulty in end users reporting human aspect defects in software, difficulty in software engineers

understanding these defects – or even appreciating them – and little or no guidelines, techniques and tools for fixing these human aspect-related defects when they occur

- development processes and techniques that still don't sufficiently include diverse stakeholder and end user perspectives – hence the co-creational living lab approach we are trialling to address this
- deficiencies in the education of software engineers regarding human aspects of their end users and the need to carefully elicit, understand and address these – one of our final year Bachelor of Engineering in Software Engineering honors team members mentioned when working on the adaptive UI components *"... someone mentioned accessibility issues once, I think, a few years ago in a UX lecture..."*

Finally, software engineers and software engineering teams themselves have many human aspects that impact how they work, think about their work, and think about their software end users, including their end user human aspects. Software engineers are humans themselves and thus are impacted by all of the human aspects end users have, to greater or lesser degree. Exactly how these software developer human aspects and end user human aspects interplay has not yet been researched in any way to our knowledge (Hidellaarachchi et al., 2021). It may be the case that various differences in developer and end user human aspects have significant impact on how the later issues are addressed (or not) in produced software.

8 SUMMARY

I have described the need to better take into account human aspects of end users during software development. This includes diverse age, ethnicity, gender, personality, emotional reactions, engagement, socio-economic status, language and language skills, culture, physical and mental challenges, cognitive problem solving style, and likely many more. These diverse human aspects of users can have a very significant impact on the usability and fit-for-purpose of the software. Currently we lack even a taxonomy to characterise and discuss such diverse human aspects of software end users. I have outlined some of our projects to address end user human aspects during requirements engineering, design, model-driven engineering, implementation, defect reporting and defect fixing. I have described our co-creational living lab approach aimed at having end users be fully involved during the whole development process for future software systems. This will, we hope, ensure their diverse human aspects are fully supported in future software systems.

ACKNOWLEDGEMENTS

Efforts of the HumaniSE team, including Lisa McGivern, Bran Salic, Anuradha Madugalla, Jen McIntosh, Ingo Mueller, Hourieh Khalajzadeh, Tanjila Kanij, Waqar Hussain, Humphrey Obie, Mojtaba Shahin and Rashina Hoda on these and related projects are greatly acknowledged. Several of these project outcomes have been produced by final year project teams, Masters and and PhD students whose enthusiasm for human-centric sofwtare engineering is wonderful to see. Aspects of this work have been supported by ARC Discovery Projects DP200100020, DP170101932 and DP140102185, Industry Transformation Research Hub IH170100013, and ARC Laureate Fellowship FL190100035.

REFERENCES

Ahmed, A. (2007). Open access towards bridging the digital divide–policies and strategies for developing countries. *Information Technology for Development*, 13(4):337–361.

Ali, N. M., Hosking, J., and Grundy, J. (2013). A taxonomy and mapping of computer-based critiquing tools. *IEEE Transactions on Software Engineering*, 39(11):1494–1520.

Alsanoosy, T., Spichkova, M., and Harland, J. (2019). Cultural influence on requirements engineering activities: a systematic literature review and analysis. *Requirements Engineering*, pages 1–24.

Alshayban, A., Ahmed, I., and Malek, S. (2020). Accessibility issues in android apps: state of affairs, sentiments, and ways forward. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 1323–1334. IEEE.

Andersen, T. O., Bansler, J. P., Kensing, F., and Moll, J. (2017). From prototype to product: Making participatory design of mhealth commercially viable. *Stud Health Technol Inform*, 233:95–112.

Avazpour, I., Grundy, J., and Grunske, L. (2015). Specifying model transformations by direct manipulation using concrete visual notations and interactive recommendations. *Journal of Visual Languages & Computing*, 28:195–211.

Avazpour, I., Grundy, J., and Zhu, L. (2019). Engineering complex data integration, harmonization and visualization systems. *Journal of Industrial Information Integration*, 16:100103.

Barnett, T., Pearson, A. W., Pearson, R., and Kellermanns, F. W. (2015). Five-factor model personality traits as predictors of perceived and actual usage of technology. *European Journal of Information Systems*, 24(4):374–390.

Burnett, M., Stumpf, S., Macbeth, J., Makri, S., Beckwith, L., Kwan, I., Peters, A., and Jernigan, W. (2016). Gendermag: A method for evaluating software's gender inclusiveness. *Interacting with Computers*, 28(6):760–787.

Carcedo, M. G., Chua, S. H., Perrault, S., Wozniak, P., Joshi, R., Obaid, M., Fjeld, M., and Zhao, S. (2016). Hapticolor: Interpolating color information as haptic feedback to assist the colorblind. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 3572–3583.

Cruz, S., da Silva, F. Q., and Capretz, L. F. (2015). Forty years of research on personality in software engineering: A mapping study. *Computers in Human Behavior*, 46:94–113.

Curumsing, M. K., Fernando, N., Abdelrazek, M., Vasa, R., Mouzakis, K., and Grundy, J. (2019). Emotion-oriented requirements engineering: A case study in developing a smart home system for the elderly. *Journal of Systems and Software*, 147:215–229.

Fensel, A., Tomic, D. K., and Koller, A. (2017). Contributing to appliances? energy efficiency with internet of things, smart data and user engagement. *Future Generation Computer Systems*, 76:329–338.

Garvie, C. and Frankle, J. (2016). Facial-recognition software might have a racial bias problem. *The Atlantic*, 7.

Grundy, J. (2020). Human-centric software engineering for next generation cloud-and edge-based smart living applications. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pages 1–10. IEEE.

Grundy, J. and Grundy, J. (2013). A survey of australian human services agency software usage. *Journal of technology in human services*, 31(1):84–94.

Grundy, J. and Hosking, J. (2002). Developing adaptable user interfaces for component-based systems. *Interacting with computers*, 14(3):175–194.

Grundy, J., Hosking, J., and Mugridge, W. (1998). Supporting large-scale end user specification of workflows, work coordination and tool integration. *Journal of Organizational and End User Computing (JOEUC)*, 10(2):38–48.

Grundy, J., Khalajzadeh, H., and Mcintosh, J. (2020). Towards human-centric model-driven software engineering. In *ENASE*, pages 229–238.

Grundy, J., Mouzakis, K., Vasa, R., Cain, A., Curumsing, M., Abdelrazek, M., and Fernando, N. (2018). Supporting diverse challenges of ageing with digital enhanced living solutions. In *Global Telehealth Conference 2017*, pages 75–90. IOS Press.

Grundy, J. and Zou, W. (2004). Auit: Adaptable user interface technology, with extended java server pages. *Multiple User Interfaces. John Wiley & Sons, New York*, pages 149–167.

Haggag, O., Haggag, S., Grundy, J., and Abdelrazek, M. (2021). Covid-19 vs social media apps: Does privacy really matter? *2021 International Conference on Software Engineering*.

Hazzan, O. and Tomayko, J. E. (2005). Reflection and abstraction in learning software engineering's human aspects. *Computer*, 38(6):39–45.

Hidellaarachchi, D., Grundy, J., Hoda, R., and Madampe, K. (2021). The effects of human aspects on the requirements engineering process: A systematic literature review. *IEEE Transactions on Software Engineering*, (01):1–1.

Hirsch, C., Hosking, J., and Grundy, J. (2010). Vikibuilder: end-user specification and generation of visual wikis. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, pages 13–22.

Hoda, R., Salleh, N., and Grundy, J. (2018). The rise and evolution of agile software development. *IEEE software*, 35(5):58–63.

Jim, A., Shim, H., Wang, J., Wijaya, L., Xu, R., Khalajzadeh, H., Grundy, J. C., and Kanij, T. (2021). Improving the modelling of human-centric aspects of software systems. In *16th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE2021), online, 26-27 April, 2021.*

Kanij, T., Merkel, R., and Grundy, J. (2015). An empirical investigation of personality traits of software testers. In *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 1–7. IEEE.

Khalajzadeh, H., Simmons, A. J., Abdelrazek, M., Grundy, J., Hosking, J., and He, Q. (2020). An end-to-end model-based approach to support big data analytics development. *Journal of Computer Languages*, 58:100964.

Khambati, A., Grundy, J., Warren, J., and Hosking, J. (2008). Model-driven development of mobile personal health care applications. In *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, pages 467–470. IEEE.

Kumar, J. (2013). Gamification at work: Designing engaging business software. In *International conference of design, user experience, and usability*, pages 528–537. Springer.

Lenberg, P., Feldt, R., and Wallgren, L. G. (2015). Behavioral software engineering: A definition and systematic literature review. *Journal of Systems and software*, 107:15–37.

Lopez-Lorca, A. A., Miller, T., Pedell, S., Mendoza, A., Keirnan, A., and Sterling, L. (2014). One size doesn't fit all: diversifying" the user" using personas and emotional scenarios. In *Proceedings of the 6th International Workshop on Social Software Engineering*, pages 25–32.

Madampe, K., Hoda, R., and Grundy, J. (2020). Towards better understanding of agile teams through behavior change models. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, pages 53–56. IEEE.

McElroy, J. C., Hendrickson, A. R., Townsend, A. M., and DeMarie, S. M. (2007). Dispositional factors in internet use: personality versus cognitive style. *MIS quarterly*, pages 809–820.

McManus, J. (2004). A stakeholder perspective within software engineering projects. In *2004 IEEE International Engineering Management Conference (IEEE Cat. No. 04CH37574)*, volume 2, pages 880–884. IEEE.

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

Newman, P., Ferrario, M. A., Simm, W., Forshaw, S., Friday, A., and Whittle, J. (2015). The role of design thinking and physical prototyping in social software engineering. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 487–496. IEEE.

Nouwen, M., Van Mechelen, M., and Zaman, B. (2015). A value sensitive design approach to parental software for young children. In *Proceedings of the 14th International Conference on Interaction Design and Children*, pages 363–366.

Obie, H. O., Hussain, W., Xia, X., Grundy, J., Li, L., Turhan, B., Whittle, J., and Shahin, M. (2021). A first look at human values-violation in app reviews. *2021 International Conference on Software Engineering*.

Osama, M., Zaki-Ismail, A., Abdelrazek, M., Grundy, J., and Ibrahim, A. (2020). Score-based automatic detection and resolution of syntactic ambiguity in natural language requirements. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 651–661. IEEE.

Perez, C. C. (2019). *Invisible women: Exposing data bias in a world designed for men*. Random House.

Rashid, Z., Melià-Seguí, J., Pous, R., and Peig, E. (2017). Using augmented reality and internet of things to improve accessibility of people with motor disabilities in the context of smart cities. *Future Generation Computer Systems*, 76:248–261.

Roturier, J. (2015). *Localizing Apps: A practical guide for translators and translation students*. Routledge.

Shamsujjoha, M., Grundy, J. C., Li, L., Khalajzadeh, H., and Lu, Q. (2021). Human-centric issues in ehealth app development and usage: A preliminary assessment. In *28th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER '21), ERA Track, Online, 9-12 March, 2021*. IEEE.

Sierra, J. S. and Togores, J. (2012). Designing mobile apps for visually impaired and blind users. In *The Fifth international conference on advances in computer-human interactions*, pages 47–52. Citeseer.

Stock, S. E., Davies, D. K., Wehmeyer, M. L., and Palmer, S. B. (2008). 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.

Strengers, Y. and Kennedy, J. (2020). *The Smart Wife: Why Siri, Alexa, and Other Smart Home Devices Need a Feminist Reboot*. MIT Press.

Taveter, K., Sterling, L., Pedell, S., Burrows, R., and Taveter, E. M. (2019). A method for eliciting and representing emotional requirements: Two case studies in e-healthcare. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pages 100–105. IEEE.

Williams, D., Alam, M. A. U., Ahamed, S. I., and Chu, W. (2013). Considerations in designing human-computer interfaces for elderly people. In *2013 13th International Conference on Quality Software*, pages 372–377. IEEE.

Winter, E., Forshaw, S., and Ferrario, M. A. (2018). Measuring human values in software engineering. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–4.

Woodson, W. E., Tillman, B., and Tillman, P. (1992). *Human factors design handbook: information and guidelines for the design of systems, facilities, equipment, and products for human use*.

Yusop, N. S. M., Grundy, J., Schneider, J.-G., and Vasa, R. (2018). Preliminary evaluation of a guided usability defect report form. In *2018 25th Australasian Software Engineering Conference (ASWEC)*, pages 81–90. IEEE.

Yusop, N. S. M., Grundy, J., Schneider, J.-G., and Vasa, R. (2020). A revised open source usability defect classification taxonomy. *Information and software technology*, 128:106396.

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

Zhao, D., Xing, Z., Chen, C., Xu, X., Zhu, L., Li, G., and Wang, J. (2020). Seenomaly: Vision-based linting of gui animation effects against design-don't guidelines. In *42nd International Conference on Software Engineering (ICSE'20). ACM, New York, NY*.