20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing - Keynote address, Nov 2-5 2020, Melbourne, Victoria, Australia (c) IEEE 2020

Human-centric Software Engineering for Next Generation Cloud- and Edge-based Smart Living Applications

John Grundy
Faculty of Information Technology, Monash University
Clayton, Victoria 3800, Australia
john.grundy@monash.edu

***Abstract*—Humans are a key part of software development, including customers, designers, coders, testers and end users. In this keynote talk I explain why incorporating human-centric issues into software engineering for next-generation applications is critical. I use several examples from our recent and current work on handling human-centric issues when engineering various ‘smart living’ cloud- and edge-based software systems. This includes using human-centric, domain-specific visual models for non-technical experts to specify and generate data analysis applications; personality impact on aspects of software activites; incorporating end user emotions into software requirements engineering for smart homes; incorporating human usage patterns into emerging edge computing applications; visualising smart city-related data; reporting diverse software usability defects; and human-centric security and privacy requirements for smart living systems. I assess the usefulness of these approaches, highlight some outstanding research challenges, and briefly discuss our current work on new human-centric approaches to software engineering for smart living applications.**

***Keywords*-smart homes; smart cities; internet of things; edge computing; human factors; diversity and inclusion; software development**

I. Introduction

Smart living applications are increasingly in demand. These range from smart homes – for general-purpose use or specific tasks e.g. supporting aged care, disability or rehabilitation; smart transport systems; smart grid and other utility-oriented systems; autonomous vehicles; ‘smart living’ robotics; industry 4.0-supporting applications; smart hospitals and schools; and smart buildings in general [1]–[3]. All of these applications require heavy use of Internet of Things-type sensors, interactors, controllers, etc [4]. All of them capture and use diverse types and amounts of data [5]. All of them increasingly require some hybrid form of cloud- and edge-computing [6]. Designing, building, deploying and maintaining such smart living solutions is technically very challenging and many research and practice issues remain to be solved. For example, Figure 1 shows two exemplar smart living solutions, a smart home to support ageing people living in their homes, and a smart urban environment. I describe these in detail below.

What often gets lost in the development of such systems are the critical human factors – what I in this keynote describe as ”human-centric issues” – relating to different aspects of the system development and usage [3], [7]. Human factors impact developing and deploying future smart living solutions in several ways: by definition, such smart living solutions often have complex end user needs. For example, end users who are aged, have a wide range of physical and mental challenges, have differing emotional reactions to new technologies, and have diverse language, cultural and socio-economic backgrounds. Development teams are often very diverse, increasingly distributed, and have differing personalities, work culture, engagement, commitment, gender and age. Organisations deploying and maintaining smart living technologies are increasingly complex, including multiple organisations with differing values having to co-operate, different members of organisations having different priorities and agendas, concerns over privacy of individuals and groups, and increasing concerns over biased ‘intelligent’ solutions.

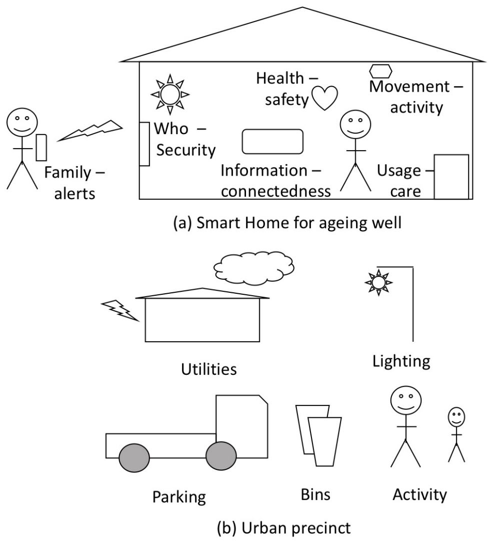


Fig. 1. Two smart living solution examples

In this keynote I outline some of these human-centric issues, the importance of fully taking them into account during systems development, and some of the serious implications of failing to properly accomodate them. I discuss a number of works of my team in trying to address some of these issues. These range from: use of human-centric, domain specific visual languages to support diverse stakeholder modelling and production of software; emotion-oriented requirements engineering, to incorporate positive and negative emotional reactions to emerging technologies into their design and evaluation; impact of personality on software developers, teams and organisations; the need for more human-centric usability defect reporting for smart living solutions; approaches to modelling complex fog- and edge-computing solutions and the need to better accomodate diverse human usage of edge devices; human-centric visualisation of smart city-related data; and the need for greatly improved security engineering and privacy models for smart technologies.

I use a few exemplar smart living solutions as motivating scenarios and in illustrating some of our work to date. These include smart homes and apps to support better ageing; smart traffic analysis and visualisation; smart city applications; and smart buildings. I discuss some outstanding issues relating to these different areas of smart living system development and deployment. Finally I outline our current approaches to improving the development of such systems. This includes use of the co-creational living lab concept, improved human-centric requirements engineering, augmenting model-driven development with human-centric issues, and deployment and evaluation with real-world smart living technology developers.

II. Motivation

A. Human-centric Issues

A wide variety of human-centric issues impact both the development of software and its usage, particularly in the smart living systems domain. Below I summarise some, but by no means all of, some of the key issues we are interested in, have worked on, or are working on currently.

Personality: Personality impact on software developers has been researched for many years in software engineering and computer science [8]. Researchers have studied personality impact on programming, testing, design, requirements engineering and maintenance. Research has shown how differing personalities of developers and team members can significantly impact software development. Much less researched to date has been on impact of personality of different end users on e.g. smart homes and building applications.

Gender: It has been long recognised that software engineering and computer science have been highly unbalanced in gender in terms of development teams, students, and researchers. Recently work has investigated how software, and other systems, are gender biased in various ways. The GenderMag toolkit provides a way to evaluate and re-design interfaces to address gender bias [9]. Several prominent mainstream articles and books have highlighted the gender bias in vehicles e.g. seat belt design, apps, and smart technologies [10].

Emotions: Different people react differently to technology solutions from an emotional perspective. Some react positively, while others negatively, to the exact same solution, which can dramatically impact the acceptance and usage of the software [11]. Software engineering researchers have become interested in emotional reactions of developers and teams under stressful situations, and how these differ between people [12].

Engagement: People engage with the same technology solutions to different levels and in different ways. This can be seen in eHealth apps and smart homes in particular. How to design such solutions to achieve high levels of engagement and usage is still very much an open research challenge [13].

Entertainment: People are highly driven by enjoyment, entertainment and ‘fun’ aspects of using software - computer games and gamification being key examples. How to better design smart living solutions to appeal to this sense of enjoyment, to better engage, attract, encourage positive emotions, and ensure take-up, are also continuing research efforts [14].

Ethnicity and Culture: Software that fails to take into account or is biased in terms ethnicity of people is highly problematic, especially for many emerging smart city applications e.g. policing and surveillance [15]. Culture is a term that incorporates a wide variety of beliefs and behaviours that need to be carefully understood and designed into smart living solutions [16]. Cultural differences also significantly impact how software development is undertaken [17].

Age: Many smart living systems focus on supporting ageing people, but some are also targeted to supporting young children [3]. People of differing ages may have quite different expectations, challenges and reactions to smart living environments, which need to be carefully designed into solutions.

Values: Human values include concepts such as inclusiveness, transparency, privacy, openness, equality and so on [18]. Many software systems conflict with one or more human values, causing expectation mis-matches and reducing usage, take-up and acceptance [19].

Physical and Mental Challenges: Many humans live with challenges including mental health, cognitive impairment and a wide variety of physical challenges, including to mobility, sight, speech, health and so on [3]. Many smart living solutions have been developed to assist with these challenges. However, all smart living systems must take into account diverse physical and mental challenges of end users in their design [20].

B. Need for IoT, Cloud and Edge Computing

All smart living applications make extensive use of Internet of Things (IoT) devices, edge computing concepts and technologies, and cloud-based compute and data services [21]. IoT devices provide an ever-growing range of technologies for sensors, interactors and controllers, heavily leveraged by smart living systems. These include, but are no means limited to: cameras, lidar, motion and movement sensors, proximity sensors, RFIDs, temperature, humidity, light, weight sensors; tablet, mobile, wearable, touch, voice, gesture, haptic, augmented and virtual reality displays and interactors; and door, light, power, water, window, network, appliance, machine, vehicle, and building controllers. The ever-growing number, data size and desire for end point computation means edge computing concepts need to be employed to achieve scalability, throughput, reliability and evolution [22], [23]. There is still a need for large-scale, multi-tenant cloud computing applications to provide compute and data services, provide building, organisation and city-wide data storage and analysis, and to off-load complex computation.

C. *Scenario 1: Smart Home for Ageing*

Consider the example from Figure 1 of a 'smart home' to assist ageing people. Such a smart living solution aims to provide ageing people with support for physical and mental challenges as they age, but want to stay in their own home longer and be safe and secure [7]. To develop a solution the software team must deeply understand technologies like sensors, data capture and analysis, communication with hospital systems, and software development methods and tools. However, they must *also* understand and appreciate the human aspects of their stakeholders: ageing people, family, and clinicians. These include the technology proficiency and acceptance of ageing people, who are likely to be much older than the software designers. The usability of the software for the very broad user base e.g. need to provide voice or gesture or smart phone interface. The emotional, both positive and negative, reactions to such a smart home e.g. they may like daily interaction with the smart home, but dislike the sense of always being monitored. The accessibility of the solutions for people with e.g. physical tremors, poor eyesight, wheel-chair bound, and cognitive decline are all highly important. Developers of the smart home must accommodate human-centric issues including diverse age, gender, culture and language of users e.g. appropriate use of language, colours, symbols. Personality differences may be very important e.g. those wanting flexible dialogue with the smart home, compared to those needing much more directive interaction with the system. Finally, the software team has many human-centric issues relating to understanding users themselves – age, language, culture, personality – e.g. over one quarter of the elderly in Australia are non-native English speakers and the majority women, but by far the majority of software developers are 20 or 30-something year old English-speaking men.

D. *Scenario 2: Urban Usage Analysis*

Smart cities have become a greatly increasing area of research but also of practice [13], [16], [20]. Consider the example from Figure 1 of a smart city solution which includes diverse data feeds to assist urban precinct operation and strategic planning. A local government instruments a wide variety of its artefacts to assist in providing services and in overall planning and management of services. These might include smart parking, lighting, rubbish bins, space usage, traffic flow, pedestrian activity, building usage, park management, utility management, and so on. Citizens are impacted by these technologies in many human-centric ways. Some citizen's values may be compromised e.g. their desire for privacy conflicts with the local government desire for openess and transparency. Some citizens may react positively to the idea of improved services via smart technologies, while others feel negatively about intrusiveness, lack of personal touch, and concern about data sharing. Some council employees find the new systems more engaging and appealing than traditional ERP systems, while others find the attempt at gamification of services annoying and distracting. While some interfaces have been carefully designed, they suffer from aspects of gender, cultural and language bias and deficiency. Surveillance mechanisms are not sufficiently broadly trained resulting in racial, age and gender biases. Finally, many citizens lack full-time access to internet services and lack sufficient educational attainment to understand complex language used in interfaces, resulting in socio-economic and educational biases.

E. *Scenario 3: Software Development Team*

To realise the previous two scenarios, a suitable software team must be formed to capture and analyse requirements, design and build systems, evaluate solutions, and deploy and evolve the smart living environments. The team has a number of human-centric challenges itself. How do team members interact and work effectively together, taking into account their disparate age, culture, language, gender, emotions, personalities, etc, especially if its a global software engineering team [24]? Given the software team is likely to be very different to target end users, how do they effectively interact with these users to capture, design and evaluation solutions? As users of these smart living systems are very diverse, how can either multiple solutions for different user groups be produced, or run-time adaptive systems be produced that can learn and adapt [25]? Ultimately, how do we move from a 'us-vs-them' model of software development to a truely co-creational smart living solution development approach [26]?

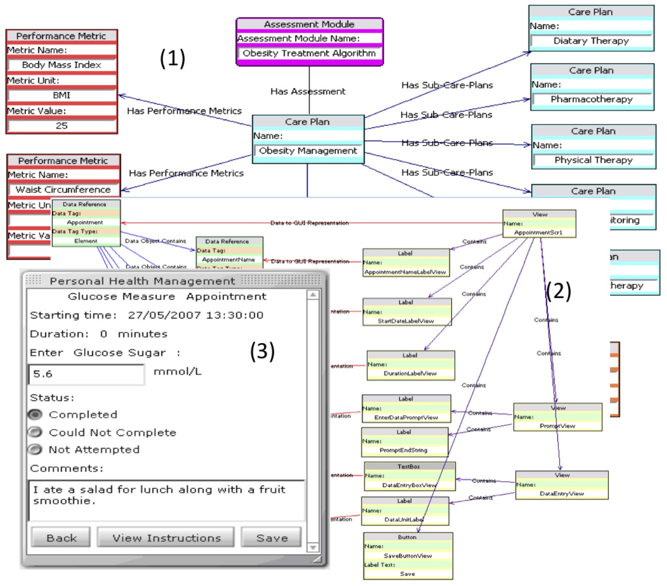
III. Examples

I describe some research projects where we have attempted to address human-centric issues highlighted in the previous section. I highlight key as yet unanswered challenges in each.

A. *Domain Specific Visual Languages*

We developed the Visual Care Plan Modelling Language (VCPML) to provide a Domain-Specific Visual Language (VCPML) approach to designing and generating mobile apps for ”care plan” implementation for diverse e-health application domains [27]. VCPML allows clinicians to model abstract, complex 'care plans' for their patients, instantiate a care plan for a specific patient and tailor it to the patient's needs, and generate a fully-functioning app from the model using model-driven engineering. Figure 2 shows an example of using VCPML to build an app for diabetes management support. (1) the clinician models the abstract care plan including exercise, pharmacology, diet, monitoring protocols, etc. (2) After specialising a generic care plan to a particular client, the clinician models desired app features using a second DSVL. (3) An app specialised to the patient care plan is generated.

While this approach is theoretically powerful, it turned out to have several severe limitations around human-centric issues of tool users and generated app users. The app interface visual notations were too complex for the clinical tool users and used terms beyond their experience and expertise.The generated



Personal Health Management

Glucose Measure Appointment

Starting time: 27/05/2007 13:30:00

Duration: 0 minutes

Enter Glucose Sugar :

5.6 mmol/L

Status:

Completed

Could Not Complete

Not Attempted

Comments:

I ate a salad for lunch along with a fruit smoothie.

Back | View Instructions | Save

Fig. 2. VCPML app design and generation example (from [27])

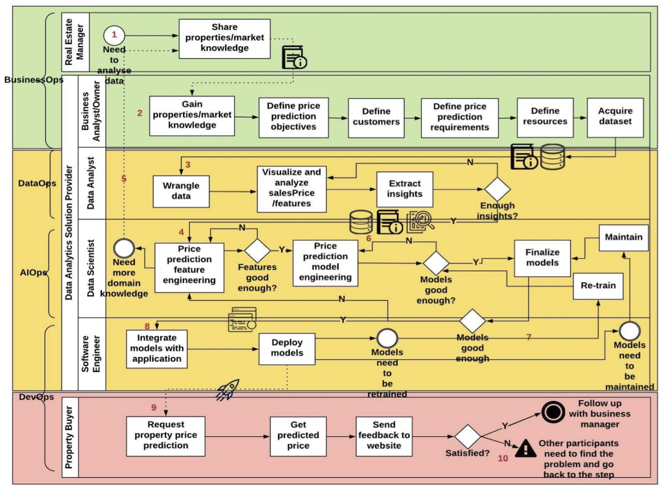

BusinessOps
Real Estate Manager
Need to analyse data
Share properties/market knowledge
Business Analyst/Owner
Gain properties/market knowledge
Define price prediction objectives
Define customers
Define price prediction requirements
Define resources
Acquire dataset
DataOps
Data Analytics Solution Provider
Data Analyst
Wrangle data
Visualize and analyze salesPrice /features
Extract insights
Enough insights?
AIOps
Data Scientist
Need more domain knowledge
Price prediction feature engineering
Features good enough?
Price prediction model engineering
Models good enough?
Finalize models
Maintain
Re-train
Software Engineer
Integrate models with application
Deploy models
Models need to be retrained
Models good enough
Models need to be maintained
DevOps
Property Buyer
Request property price prediction
Get predicted price
Send feedback to website
Satisfied?
Follow up with business manager
Other participants need to find the problem and go back to the step

Fig. 3. BiDaML Example (from [28])

apps could not be tailored to meet the needs of diverse end users. The language used in the generated apps assumed too great an education level and English language competence; the apps used fiddly icons and interactors not suitable for many of the ageing users; the apps assumed a Euro-centric view of health and suffered from gender-biased choices; the apps provided open dialogue with clinical staff but with the information sharing violating many patients' values around privacy, openness and informed consent.

Many smart living applications require complex data capture, processing and visualisation. Most teams working on such applications are diverse: data scientists, end users, business sponsors, domain experts, software engineers and cloud platform specialists. Unfortunately there does not exist a set of high level visual modelling languages to support these diverse

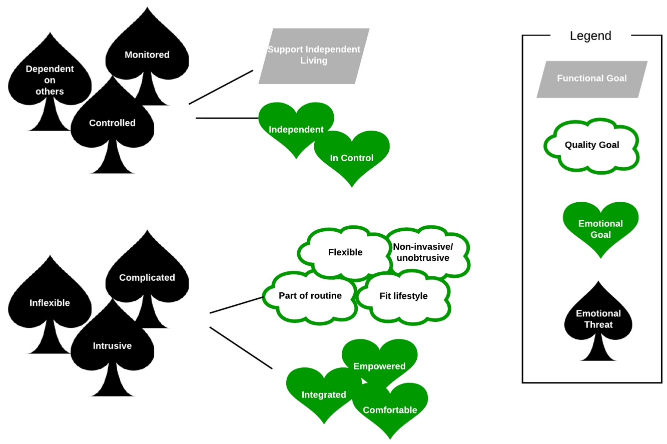


Fig. 4. Emotion-oriented RE for smart home design example

data analytics application teams to be able to effectively work together [28]. To address this gap we developed the Big Data Analytics Modelling Languages (BiDaML) to provide a set of DSVLs to support diverse teams working on complex data analytics applications. BiDaML provides several diagram types for this purpose: Brainstorming diagrams provide an overview of a data analytics project and all the tasks and sub-tasks involved in designing the solution; Process diagrams specify the analytics processes/steps including key details related to the participants, operations, and data items, capturing details from a high-level to a lower-level; Technique diagrams show the step by step procedures and processes for each sub-task at a low level of abstraction; Data diagrams document the data and artefacts that are produced in each of the above diagrams; and Output diagrams define the outputs associated with different tasks e.g. output information, reports, results, visualisations, etc. Figure 3 shows a Process Diagram from BiDaML. This shows an example from a real-world property price analysis problem where several datasets need to be collected, aggregated, analysed and property price predictions visualised for real estate analysts. The team developing this solution includes banking and real estate experts, data scientists, software engineers and cloud platform host engineers.

While our BiDaML approach has proved very effective as a human-centric set of DSVLs for this domain, it still lacks support for many human-centric end user issues. We can not capture different human-centric needs of end users including age, language, culture, educational level, socio-economic status, gender etc which may impact appropriateness and usability of the produced solution. When used in smart living systems, we may wish to model and leverage different user engagement, enjoyment, motivation, personality and values differences. We may wish to produce a variety of solutions based on some of these diverse end user characteristics, or a single solution that can adapt or be adapted to these user needs.

B. *Emotion-oriented Requirements Engineering*

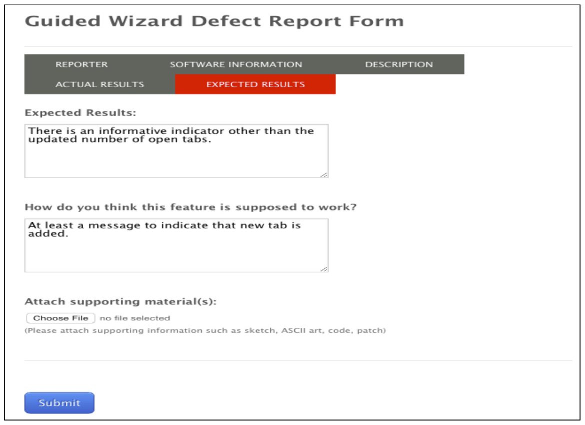
We have applied an augmented DSVL [11] to support the modelling of emotions in a variety of smart living systems [3]. The idea is to capture goal models representing key requirements of the target systems but augmented with 'positive' and 'negative' emotions that the systems may elicit in end users. Developers and stakeholders can the reason about ways to mitigate the negative emotions some users may face, while strengthening positive reactions for all users. Figure 4 shows an example of part of an emotion-oriented requirements goal model. In this model, a conventional goal-directed requirements modelling language has been augmented to allow requirements engineers to capture and relate possible emotional goals (typically positive emotions to try and achieve) and threats (typically negative emotional reactions). In this example we see the target system functional goal of supporting independent living in the smart home may be threatened by negative senses of over-dependency, monitoring and control. The goal is however supported by positive senses of maintaining independence and self-control. The requirements engineers want developers to try and mitigate the former while achieving/strengthening the later in their smart home design. An associated evaluation framework enables the team to test the implementation against these emotional goals.

While this model has proved useful for aged care-oriented smart living solutions, it still has numerous challenges. We need a way of ensuring these emotions are linked to design and implementation decisions and features. We need to link emotional goals to other human-centric goals e.g. personality, gender, culture, values etc. We need to provide developers with tools to test and report emotion-oriented goal failures in the system. Finally, we need to be able to model many other human-centric requirements issues that may be impacted in the development of the smart living solution, without overwhelming models with complexity. We investigated age, gender and physical challenges implicitly in this work, as the personas representing stakeholders needed to capture these. But this was not done and evaluated systematically, nor was support for modelling these human-centric characteristics in a similar way to emotions undertaken.

C. *Personality and Software Teams*

Personality and software development has long been studied [8]. We have been interested in several aspects of this problem: personality and its impact on different phases of development, specifically software testing [29]; personality and its impact on learning pair-programming [30]; personality and its impact on requirements engineering [31]; and personality and its impact on teams and organisations [32]. These have shown that different aspects of software activity can be significantly impacted by different developer, pair, and team personalities.

Less studied is personality impact on the resulting smart living solutions themselves, and the personality interactions of developers and stakeholders. Both of these need to be further investigated to better understand how, like different emotional reactions to technology solutions, they impact on smart living system take-up and usage. It is likely that people with different personalities will find different solutions better – or worse – for supporting their interactions than others. It is likely that some interactions between requirements engineers and stakeholders will be significantly impacted by personality differences [33].


Guided Wizard Defect Report Form

Expected Results:

There is an informative indicator other than the updated number of open tabs.

How do you think this feature is supposed to work?

At least a message to indicate that new tab is added.

Attach supporting material(s):

Choose File no file selected

(Please attach supporting information such as sketch, ASCII art, code, patch)

Fig. 5. Usability Defect Reporting (from [34])

D. *Usability Defects and Reporting*

Software systems released to end users often have severe usability defects [35]. While much work has been done in HCI to investigate usability engineering approaches, much less has been done in software engineering research or translated to software engineering practice. Most defects are reported via general-purpose issue tracking tools like JIRA and Bugzilla. We have been working on improving usability defect reporting via an improved defect taxonomy and structured reporting form [34]. Figure 5 shows a prototype of such a form in use to report a defect. The approach uses a wizard-based metaphor, stepping the reporter through a set of question answering steps to represent the usability defect in detail.

While this approach is more effective than using traditional defect reporting tools, many outstanding issues remain. We have focused on web and mobile applications, but reporting IoT-based sensor, interactor and controller defects will be important for most smart living systems. Existing usability defect taxonomies do not incorporate even smart phone/tablet-based interfaces, let alone such IoT-based interfaces [35]. Almost no work yet seems to exist on ways to effectively report usability – or other human-centric defects – in smart home/smart city type applications [36].

E. *Deploying Large Edge-based Applications*

Most smart living solutions need to be deployed on edge or fog computing systems. These turn out to have a number of significant differences to deploying them on cloud-based systems [22]. This includes the use of a very large number of edge end points, the heterogeneity of the edge devices and edge servers; the need to distribute large amounts of compute and data; the need to achieve high levels of reliability and robustness, due to the critical application domain nature of many smart living systems; and the need to accomodate new edge

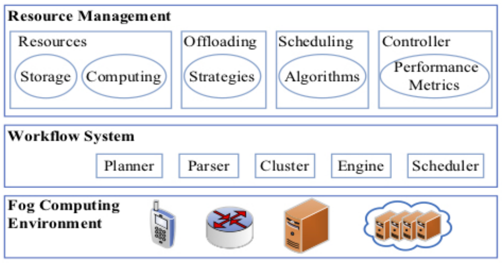


Fig. 6. Fog Workflow Simulator (from [37])

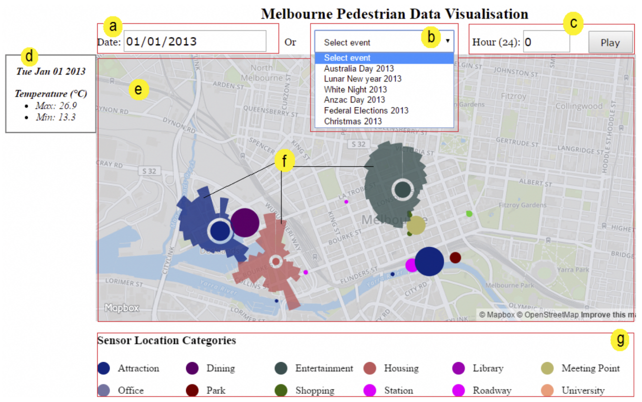


Fig. 7. PedaViz Example (from [40])

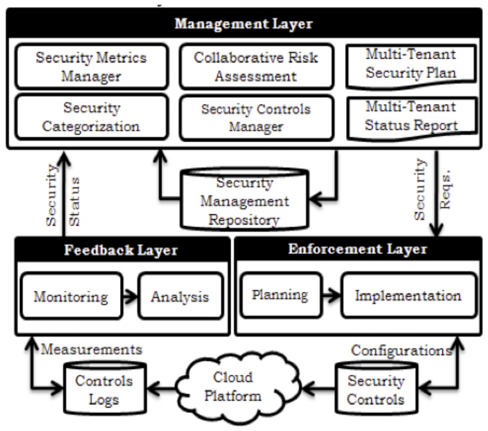


Fig. 8. Collaborative Security Example (from [42])

devices and services during system operation. Understanding how to effectively deploy applications to edge computing platforms is not well understood.

We have been investigating several aspects to this edge deployment challenge. We developed a toolkit, FogWorkflowSim, to provide developers and deployers a way to experiment with performance of complex workflow systems deployed to an edge computing domain [37]. Figure 6 shows the architecture of this toolkit. The engineer can define available fog (edge) computing resources, including storage and compute, for different devices and servers. They can choose different offloading strategies, scheduling algorithms and choose different performance metrics of the solution of interest e.g. throughput, number of users, cost. The workflow system defined will be spread across the defined fog platform including edge devices, edge services, cloud compute and cloud data. We have also been investigating the impact of caching and compute distribution strategies for edge computing applications [38], [39]. We want improved algorithms to deploy applications to large scale edge systems, and includes optimising compute and data caching for different users, edge devices and servers, cloud services, and applications.

To date in this work we have taken little account of different edge device user human-centric requirements. For example, in a smart building domain we may have greatly varied end users in terms of data privacy, security, transparency and openness requirements. We may have greatly varied application edge device usage characteristics, based on user age, emotions, values, or physical or mental challenges. User mobility and context change of devices may be greatly influenced by these and other human characteristics of smart living system users.

F. Visualising Smart City Application Data

Once we have deployed a smart living solution to an edge platform, we need to capture diverse data, aggregate the data, analyse it, and visualise various resultant information. In our PedaViz work [40], we developed a tool based on urban precinct usage data, captured from a variety of pedestrian tracking devices, to better understand short term and longer term precinct activities. Figure 7 shows an example of this tool in use on Melbourne City Council pedestrian data. Here we visualise activity over a 24 hour period using a novel map + clock overlap visualisation technique. Council managers can better understand how pedestrian usage changes during the day, due to weather impacts, due to cultural or sporting events, or due to other activities occurring nearby. We similarly combined a variety of social media data and geographic data to produce a better understanding and recommendation of visits to tourist points of interest [41].

Outstanding issues in this domain include the impact of human characteristics on the use of the visualisations, the capture of data, and their activities being tracked. Different users may have very different information visualisation tasks they wish to use the analysed data for. This may result in very different interfaces needing to be provided, based on their personality, culture, language, gender, physical challenges etc. Different users may have very different values relating to information disclosure, capture and usage. Different users may have very different activity patterns around mobility, relating to physical and mental challenges, age, socio-economic status.

G. Security Requirements for Smart Living Solutions

Smart living solutions have very diverse security and privacy requirements for their different users. In earlier work we developed a collaborative cloud security management framework and toolset [42]. This allowed cloud providers, application providers, and application users to tailor a range of security enforcement solutions to different cloud tenant needs. Figure 8 shows an outline architecture for this approach. A management layer provides run-time configurable security services including metrics, categorisation, risk assessment, configuration controls, tenant-specific plans and reporting. An enforcement layer takes configurations and applies on a per-tenant, per-application basis. A feedback later uses measurements to monitor, analyse and – via the management reporting – visualise security enforcement behaviours.

We want to apply a similar approach to – much more complex – edge-based systems for smart living domains. Besides the technical issues of developing such a solution for far more heterogeneous devices, services and applications than on a cloud platform, human-centric issues impact many aspects. A much wider group of users will need to be supported, most without any technical knowledge of security and privacy concepts. Different values, emotions, personalties and cultures will undoubtedly make the security configuration and monitoring needed far more complex [43], [44]. Diverse human-centric privacy requirements will need to be supported.

IV. Outstanding Smart Living Human-centric Engineering Challenges

I summarise some of the key outstanding challenges of engineering next-generation smart living solutions, from the perspective of the diverse human-centric issues that impact both the engineering and usage of such systems.

A. Access to and Participation of Diverse Users

Effective requirements engineering for smart living solutions requires access to, and deep involvement of, a diverse range of end users throughout the development of these systems. Actually finding and obtaining this input is more difficult than many engineers appreciate. Our software – and systems – engineering methodologies tend to foster a 'them and us' approach – engineers being us, everyone else being them. Even agile approaches with 'customer in team' practices often fail to actually listen to users, listen to enough users, and treat their concerns with the importance that they are due. To engineer effective smart living solutions, we need to completely rethink what software engineering, participatory design, agile methods, user centred design, evaluation, and project leadership and management mean in this context.

B. Human-centric Development

Such systems are built by and for humans. The dynamics of humans working together to engineer such complex smart living technologies requires much further research and practice improvements. Linked to the previous item, teams are made up of individuals with diverse personalities, experiences, cultures, language, genders, age, physical and mental challenges, human values, educational backgrounds and so on. All of these may have a greater or lesser impact on the way they work together, and work with smart living solution users, to engineer the solutions. Global software engineering projects mean different teams and organisations need to work across countries, introducing even more human-centric development complexities.

C. Capturing Human-centric Issues

Actually capturing diverse human-centric issues relating to smart living systems requires far more work. I gave examples of capturing emotions relating to potential end user reactions to a smart home system, limited examples of capturing age and language, and limited examples of capturing personality in development teams. I gave examples of domain-specific visual languages and human-centric visualisations. All of these, and others' related works in this area, only address a small part of human-centric issues impacting smart living systems. Currently no design principles exist for developing such modelling techniques, nor do any mechanisms to check correctness, completeness and consistency of many human-centric issues in requirements.

D. Developing and Deploying Adaptive Software

We need better support for engineering smart living solutions. The example of the eHealth app I used earlier illustrates the challenges we face to do this. We need to incorporate the human-centric requirements for diverse end users into not only requirements capturing modelling languages, but into architecture, design, interface, and code specifications. This may result in us wanting to produce a great many different implementations for the same smart living system. It may need us to adapt the single solution in a great many – sometimes unanticipated – ways at run-time.

E. Data Security, Privacy, Provenance

Smart city technologies by their very nature produce, capture, aggregate, analyse, present and share a very wide rage of information. Given they often fulfil a number of safety and security-critical services, they require great attention to security and privacy aspects. However, this is further complicated by the diverse human-centric issues that are critical to the success of such systems. Different users, groups of users, teams, organisations and ultimately societies have diverse human values impacting on smart living system behaviour. Different users react emotionally to the provision of such services and their data provenance and privacy implications in different ways. Cultural expectations may impact their usage, adoption and acceptance of solutions. People react very differently to cybersecurity events and privacy attacks based on personality, age, education and other factors. Failure to account for bias around ethnicity, gender, educational attainment, socio-economic status, age, physical and mental challenges and other human characteristics have all been shown to have serious implications for smart living system usage.

F. Large-scale Deployment and Evolution

Deploying smart living solutions to edge computing platforms brings a large number of technical challenges, including scalability, robustness, reliability, maintenance, evolution, and so on. In my view, the human-centric issues that these technologies must cope with are at least as challenging. Different users will have very different usage patterns, data collection and informed consent requirements, human relationships to

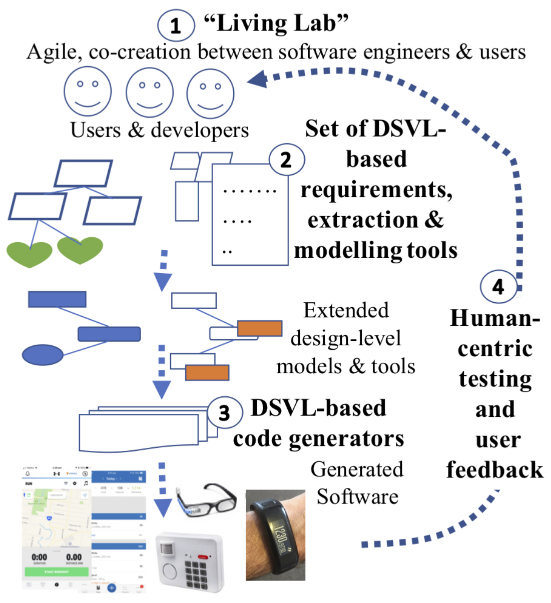

1 "Living Lab"
Agile, co-creation between software engineers & users
Users & developers
2 Set of DSVL-based requirements, extraction & modelling tools
Extended design-level models & tools
3 DSVL-based code generators
Generated Software
4 Human-centric testing and user feedback

Fig. 9. Living lab

family, friends and carers, cultural norms, language, technology competence and acceptance, and ultimately different outcomes they want and expect from these systems. As noted above, we still lack adequate development methods, modelling approaches, and engineering support for the development and deployment of these systems. Many smart living systems have emergent requirements i.e. those not anticipated at design time. This includes new technologies e.g. new devices, data, usage scenarios. But it also includes unanticipated human-centric issues e.g. support for new languages and cultural expectations; support for a wider range of aged users; support for mental challenges of users not understood by developers; use of intelligent services that may be unexpectedly biased along ethnic, gender, age, etc dimensions; and misalignment of values between developers, deploying organisations and users.

G. *Evaluating Deployed Software*

Conducting real-world experiments with real users on deployed smart living systems is essential. However, as identified above, their is almost no work so far in the area of human-centric issues defect reporting, defect analysis, defect correction and ultimately quality assurance of such systems. IoT systems in general lack appropriate approaches for managing human-centric usability engineering. These are fundamentally very different from desktop, web and mobile app solutions. Smart living systems by their very nature must cope with a wider variety of human user challenges, many that developers do not live, understand or are simply unaware of. Updates made to deployed smart living solutions must be very carefully rolled out and how previously highlighted human-centric defects are addressed, while not introducing new ones.

V. Our Approach

We are working on a new approach to engineer smart living solutions. This includes smart homes for ageing, rehabilitation and physically and mentally challenged; smart technologies to support ageing, mental health and loneliness; various smart city solutions; and smart solutions for vulnerable community members. Figure 9 illustrates the new human-centric, model-driven software engineering approach we aim to produce in our current work. (1) We are adopting an Agile Living Lab approach that co-locates the software team and target end users of smart living systems throughout the process [3]. This will provide a co-creational culture and environment to better elicit human-centric requirements, model and capture with human-centric DSVLs, and receive continuous feedback from users. (2) A set of customised DSVL tools will be used to capture and model the human-centric requirements, validate them against design principles and best practice modelling patterns, and translate them to extended design-level models. (3) A set of MDSE generators will generate smart living software applications – code, configurations, etc – for devices, servers, apps and other components. Unlike existing generators, these will take into account wide variations of end-users as specified in the human-centric requirements. These will produce either multiple versions of the target software applications and/or reconfigurable applications that adapt to each end user?s differing human-centric needs. (4) A combination of human-centric requirements testing and continuous defect feedback will be fed to the development team. By leveraging the Living Lab concept, this will enable both faster feedback and defect correction, but also better evolution and modelling of the human-centric requirements over time. Lessons will be fed into the improvement of the DSVL tools, best practice patterns and MDSE generators. We have identified a set of necessary research activities to achieve this outcome.

A. *Co-creational Living Lab*

Human-centric requirements have to be elicited from target end users and stakeholders, captured and modelled using our DVSL-based tools, used by our extended model-driven engineering solutions to generate software, and then the software tested and user feedback accepted and actioned to correct requirements and design model problems. A new approach is needed to effectively support the software team in achieving this. We are adopting the Living Lab co-creation concept that has become popular in digital health software development [3]. We are establishing this lab with a domain-specific focus with smart living technology producer and user partner companies and target end users and the software team co-located. Target end users and developers closely collaborate to elicit, capture, test, use and refine the human-centric software requirements. The DSVL modelling tools, MDSE generators and testing tools all need to support collaborative capture, discussion and refinement of the human-centric requirements for this to be most effective. We are extending our current work on developing digital health technologies [3], human-centric software engineering processes in software teams, including personality and team climate [29], [30], and collaborative DSVL-based modelling tools [45].

B. Human-centric Requirements Engineering

We are developing a range of new and augmented DSVLs to model a wide variety of human-centric issues at the requirements level for software systems. Some of these DSVLs extend existing requirements modelling languages in successively more principled ways than currently e.g. goal-directed requirements languages such as i*, use cases and essential use cases, target user personas, user stories, etc. However, others may provide wholly novel requirements modelling techniques and diagrams that are then linked to other requirements models. We envisage novel requirements capture for things like identifying cultural, age, accessibility and personality aspects of target end users. Where multiple target end users for the same software application have differing human-centric requirements, multiple or composite models will likely be necessary. We are building on a wide range of DSVLs, including for design tools, requirements, reporting, business processes, surveys, performance testing, and many others, that we have previously developed [28] as well as digital health software [3] and work on modelling usability defects and emotional and multi-lingual requirements [7], [31].

C. Human-centric Model-driven Engineering

Once we have some quality design-level human-centric issues in models (incremental outcomes from activities 5.4-5.7), we can use these in model-driven engineering code and configuration generators. Results from this work will be fed back to extending the our human-centric modelling language design principles. This will involve adding generators that consume design level models augmented with human-centric properties and synthesizing software applications that use these appropriately. For example, we might generate a gesture-based, passive-voice feedback solution for a target user who has motor challenges and anxious temperment. However, we might generate several interfaces for the same software feature, and at run-time configure the software either with pre-deployment knowledge, end user input, or even modify it while in use based on end user feedback. Thus for example a part of the software for our smart home scenario could adapt to different end users' current and changing needs (e.g. age, culture, emerging physical and mental challenges, personality etc). This work will be done incrementally, focusing on single issues first then looking at successively more complex combinations, adding support to the prototype tools and repeatedly trialling the tools. We will work on new approaches to human-centric adaptive user interfaces [46], adaptive run-time software [42], and DSVL-based MDSE solutions [45].

D. Human-centric Application Evaluation

We need to address critically important issues of (i) testing whether the resultant smart living software actually meets the requirements specified; (ii) providing a feedback mechanism for end users to report defects in their smart living systems, specifically relating to human-centric issues; and (iii) providing a feedback mechanism from software developers to users about changes made relating to their personal human-centric issues. We are developing a human-centric requirements-based testing framework, techniques and tools. These enable human-centric issues to be used in acceptance tests to improve validation of software against these requirements. We will also look to develop new human-centric defect reporting mechanisms and developer review and notification mechanisms. These will support continuous defect reporting, correction, and feedback via the living lab and remotely. This work extends our research on software tester practices and usability defect reporting [35] and requirements-based testing [47].

VI. Summary

I have highlighted the many challenges human-centric issues that present when trying to engineer next-generation 'smart living' systems. While the technological challenges of engineering these systems are considerable, they bring a wealth of very challenging human-centric issues, including but not limited to the personality, emotions, age, gender, culture, ethnicity, physical and mental challenges, values, engagement and desires of human engineers and stakeholders. I have shown that we need significantly improved development methods to address these challenges. Building on our experiences to date, I have highlighted some of the key approaches we are using to address – some of, but by no means all – of these very important human-centric issues.

Acknowledgment

Grundy is supported by an Australian Laureate Fellowship FL190100035. Aspects of this work have been supported by ARC Discovery Projects DP110101340, DP140102185 and DP170101932, ARC ITRH IH170100013, the New Zealand FRST SER Software Process and Product Improvement project, National ICT Australia, Malaysian Government, Swinburne University of Technology and the University of Auckland. I thank the many collaborators, research fellows and research assistants, and PhD and Masters students that have worked with me on this wide range of projects.

References

[1] J. Al-Jaroodi and N. Mohamed, "Service-oriented architecture for big data analytics in smart cities," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 2018, pp. 633–640.

[2] A. De Iasio, A. Futno, L. Goglia, and E. Zimeo, "A microservices platform for monitoring and analysis of iot traffic data in smart cities," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 5223–5232.

[3] J. Grundy, K. Mouzakis, R. Vasa, A. Cain, M. Curumsing, M. Abdelrazek, and N. Fernando, "Supporting diverse challenges of ageing with digital enhanced living solutions," in *Global Telehealth Conference 2017*. IOS Press, 2018, pp. 75–90.

[4] T. K. Hui, R. S. Sherratt, and D. D. Sánchez, "Major requirements for building smart homes in smart cities based on internet of things technologies," *Future Generation Computer Systems*, vol. 76, pp. 358–369, 2017.

[5] Z. Khan, A. Anjum, and S. L. Kiani, "Cloud based big data analytics for smart future cities," in *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*. IEEE, 2013, pp. 381–386.

[6] P. Wang, L. T. Yang, and J. Li, "An edge cloud-assisted cpss framework for smart city," *IEEE Cloud Computing*, vol. 5, no. 5, pp. 37–46, 2018.

[7] M. K. Curumsing, N. Fernando, M. Abdelrazek, R. Vasa, K. Mouzakis, and J. Grundy, "Understanding the impact of emotions on software: A case study in requirements gathering and evaluation," *Journal of Systems and Software*, vol. 147, pp. 215–229, 2019.

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

[9] M. Burnett, S. Stumpf, J. Macbeth, S. Makri, L. Beckwith, I. Kwan, A. Peters, and W. Jernigan, "Gendermag: A method for evaluating software's gender inclusiveness," *Interacting with Computers*, vol. 28, no. 6, pp. 760–787, 2016.

[10] C. C. Perez, *Invisible Women: Exposing data bias in a world designed for men*. Random House, 2019.

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

[12] S. C. Müller and T. Fritz, "Stuck and frustrated or in flow and happy: sensing developers' emotions and progress," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 688–699.

[13] A. Fensel, D. K. Tomic, and A. Koller, "Contributing to appliances? energy efficiency with internet of things, smart data and user engagement," *Future Generation Computer Systems*, vol. 76, pp. 329–338, 2017.

[14] J. Kumar, "Gamification at work: Designing engaging business software," in *International conference of design, user experience, and usability*. Springer, 2013, pp. 528–537.

[15] C. Garvie and J. Frankle, "Facial-recognition software might have a racial bias problem," *The Atlantic*, vol. 7, 2016.

[16] L. Mora, R. Bolici, and M. Deakin, "The first two decades of smart-city research: A bibliometric analysis," *Journal of Urban Technology*, vol. 24, no. 1, pp. 3–27, 2017.

[17] T. Alsanoosy, M. Spichkova, and J. Harland, "Cultural influence on requirements engineering activities: a systematic literature review and analysis," *Requirements Engineering*, pp. 1–24, 2019.

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

[19] J. Whittle, "Is your software valueless?" *IEEE Software*, vol. 36, no. 3, pp. 112–115, 2019.

[20] Z. Rashid, J. Melià-Seguí, R. Pous, and E. Peig, "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*, vol. 76, pp. 248–261, 2017.

[21] J. M. Schleicher, M. Vögler, S. Dustdar, and C. Inzinger, "Enabling a smart city application ecosystem: requirements and architectural aspects," *IEEE Internet Computing*, vol. 20, no. 2, pp. 58–65, 2016.

[22] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[23] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the internet of things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, 2016.

[24] H. Holmstrom, E. Ó. Conchúir, J. Agerfalk, and B. Fitzgerald, "Global software development challenges: A case study on temporal, geographical and socio-cultural distance," in *2006 IEEE International Conference on Global Software Engineering (ICGSE'06)*. IEEE, 2006, pp. 3–11.

[25] H. Song, S. Barrett, A. Clarke, and S. Clarke, "Self-adaptation with end-user preferences: Using run-time models and constraint solving," in *International Conference on Model Driven Engineering Languages and Systems*. Springer, 2013, pp. 555–571.

[26] G. Nesti, "Co-production for innovation: the urban living lab experience," *Policy and Society*, vol. 37, no. 3, pp. 310–325, 2018.

[27] A. Khambati, J. Grundy, J. Warren, and J. Hosking, "Model-driven development of mobile personal health care applications," in *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 2008, pp. 467–470.

[28] H. Khalajzadeh, M. Abdelrazek, J. Grundy, J. Hosking, and Q. He, "Bidaml: A suite of visual languages for supporting end-user data analytics," in *2019 IEEE International Congress on Big Data (BigDataCongress)*. IEEE, 2019, pp. 93–97.

[29] T. Kanij, R. Merkel, and J. Grundy, "An empirical investigation of personality traits of software testers," in *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE, 2015, pp. 1–7.

[30] N. Salleh, E. Mendes, and J. Grundy, "Investigating the effects of personality traits on pair programming in a higher education setting through a family of experiments," *Empirical Software Engineering*, vol. 19, no. 3, pp. 714–752, 2014.

[31] M. Kamalrudin, S. Sidek, N. Salleh, J. Hosking, and J. Grundy, "A pair-oriented requirements engineering approach for analysing multi-lingual requirements," in *Requirements Engineering*. Springer, 2014, pp. 150–164.

[32] A. B. Soomro, N. Salleh, E. Mendes, J. Grundy, G. Burch, and A. Nordin, "The effect of software engineers? personality traits on team climate and performance: A systematic literature review," *Information and Software Technology*, vol. 73, pp. 52–65, 2016.

[33] P. K. Murukannaiah, N. Ajmeri, and M. P. Singh, "Acquiring creative requirements from the crowd: Understanding the influences of personality and creative potential in crowd re," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*. IEEE, 2016, pp. 176–185.

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

[35] N. S. M. Yusop, J. Grundy, and R. Vasa, "Reporting usability defects: A systematic literature review," *IEEE Transactions on Software Engineering*, vol. 43, no. 9, pp. 848–867, 2016.

[36] M. Bures, T. Cerny, and B. S. Ahmed, "Internet of things: Current challenges in the quality assurance and testing methods," in *International Conference on Information Science and Applications*. Springer, 2018, pp. 625–634.

[37] X. Liu, L. Fan, J. Xu, X. Li, L. Gong, J. Grundy, and Y. Yang, "Fogworkflowsim: An automated simulation toolkit for workflow performance evaluation in fog computing," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 1114–1117.

[38] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *International Conference on Service-Oriented Computing*. Springer, 2018, pp. 230–245.

[39] X. Xia, F. Chen, Q. He, G. Cui, P. Lai, M. Abdelrazek, J. Grundy, and H. Jin, "Graph-based optimal data caching in edge computing," in *International Conference on Service-Oriented Computing*. Springer, 2019, pp. 477–493.

[40] H. O. Obie, C. Chua, I. Avazpour, M. Abdelrazek, J. Grundy, and T. Bednarz, "Pedaviz: Visualising hour-level pedestrian activity," in *Proceedings of the 11th International Symposium on Visual Information Communication and Interaction*, 2018, pp. 9–16.

[41] D. Chen, D. Kim, L. Xie, M. Shin, A. K. Menon, C. S. Ong, I. Avazpour, and J. Grundy, "Pathrec: Visual analysis of travel route recommendations," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 2017, pp. 364–365.

[42] M. Almorsy, J. Grundy, and A. S. Ibrahim, "Collaboration-based cloud computing security management framework," in *2011 IEEE 4th International Conference on Cloud Computing*. IEEE, 2011, pp. 364–371.

[43] F. Kammüller, "Human centric security and privacy for the iot using formal techniques," in *International Conference on Applied Human Factors and Ergonomics*. Springer, 2017, pp. 106–116.

[44] M. Almorsy, J. Grundy, and I. Müller, "An analysis of the cloud computing security problem"," *Proceedings of the 2010 Asia Pacific Cloud Workshop, Colocated with APSEC2010, Australia*, 2010.

[45] J. C. Grundy, J. Hosking, K. N. Li, N. M. Ali, J. Huh, and R. L. Li, "Generating domain-specific visual language tools from abstract visual specifications," *IEEE Transactions on Software Engineering*, vol. 39, no. 4, pp. 487–515, 2012.

[46] J. Grundy and J. Hosking, "Developing adaptable user interfaces for component-based systems," *Interacting with computers*, vol. 14, no. 3, pp. 175–194, 2002.

[47] M. Kamalrudin, J. Hosking, and J. Grundy, "Maramaaic: tool support for consistency management and validation of requirements," *Automated software engineering*, vol. 24, no. 1, pp. 1–45, 2017.