Integrating Social Values into Software Design Patterns

Waqar Hussain, Davoud Mougouei, Jon Whittle
Faculty of Information Technology
Monash University
Clayton, Victoria, Australia
{waqar.hussain, davoud.mougouei, jon.whittle}@monash.edu

ABSTRACT

Software Design Patterns (SDPs) are core solutions to the recurring problems in software. However, adopting SDPs without taking into account their value implications may result in breach of social values and ultimately lead to user dissatisfaction, lack of adoption, and financial loss. An example is the airline system that overcharged people who were trying to escape from the Hurricane Irma. Although not intentional, overlsight of social values in the design of the airline system resulted in significant customer dissatisfaction and loss of trust. To mitigate such value breaches in software design we propose taking social values into account in SDPs explicitly. To achieve this, we outline a collaborative framework that allows for (i) specifying the value implications of SDPs, (ii) developing or extending SDPs for integrating social values, (iii) providing guidance on the value-conscious adoption of design patterns, (iv) collecting and analyzing insights from collaborators, (v) maintaining an up-to-date library of the valufied design patterns, and (vi) incorporating lessons learned from the real-world adoption of the valuefied design patterns into the proposed framework for its continuous improvement in integrating social values into software.

KEYWORDS

Design Patterns, Social Values, Framework, Fairness

1 INTRODUCTION

Software embodies social values that are intricately intertwined, but remain masked from the common view. The far reaching economic, political, and social impact of software has long been recognized in a variety of disciplines including social sciences, engineering and health informatics [16, 23, 32]. As software gets woven deeper into the very fabric of modern societies making them increasingly dependent on large and complex software systems, designing software for societal values has become increasingly important [33]. However, value breaches in software continue to make the headlines for example the societal impact of Volkswagon fuel emission fiasco [1] and its resulting impact on the company's financial position and reputation. Similarly the debacles of Novopay and Queensland payroll systems in New Zealand and Australia respectively, where value breaches from the involved parties resulted in huge financial losses and the credibility of the concerned governments to deliver reliable systems for public use [9, 31]. Despite the availability of several Software Engineering codes of ethics [19, 20, 30] urging technologist to address societal values in technology creation, violations of values persist in various software domains such as Robotics, Machine Learning and AI [6], Software Design [35], Internet of Things, and eHealth solutions [2, 33, 34]. The perceived omissions of certain stakeholder values by the implementers as a result, has created an atmosphere of distrust which often leads to reluctance of end-users to adopt technological solutions.

The discipline of design can induce social change and has placed designers in a position to project ethical and meaningful change [35]. Designers are thus, entrusted with immense power as well as ethical and moral responsibility [4]. However, we are far from understanding the specifics of how designers' power and responsibility is enacted during the process of designing a software.

Design Patterns, as templates of captured experiential wisdom, provide empirically proven solutions to address recurring design problems [36]. Although design patterns do not deal with values explicitly, they can be used in combination with best practices from participatory design to incorporate relevant human values into technology [11]. The interrelation of technical design and social values is well understood and reported [5, 14, 15]. Design patterns are applied to implement technical design decisions at key junctures of software development which makes them instrumental for injecting social values into software. We posit that value breaches are often caused by not considering social values during software design e.g. adopting certain design patterns that either lack the desired values or are misaligned with them. To mitigate value breaches and their unintended yet adverse impacts on the society, we propose to take social values into account explicitly during design by embedding these values into design patterns, i.e. to valuefy design patterns.

To this end we propose a framework referred to as the Value-Design Hub (VDH), which allows for valuefication of software design patterns. VDH is aimed at bringing out value propositions implicit in design patterns and facilitating their specification through stakeholder collaboration. Furthermore, using VDH, the collaborative selection and adoption of the valuefied design patterns during design and development will allow for a more consistent application of social values in software technologies. For valuefication of software design patterns this research specifically aims to:

- **(A1)** Specify the value implications of the existing design patterns and the dependencies/conflicts among them;
- **(A2)** Extend the existing design patterns or develop new patterns to account for social values;
- **(A3)** Develop Guidelines, Indicators, Tools, and Techniques (GITTs) for value-conscious adoption of design patterns.

FairWare'18, May 29, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5746-3/18/05…$15.00
https://doi.org/10.1145/3194770.3194777

Valuefication of the design patterns needs to be conducted in close participation with entities involved in the design and use of software. Such valuefication must be carried out based on the insights from stakeholders such as users, software design experts, and social scientists. Similarly, it is important to monitor the adoption of the valuefied design patterns in software projects and incorporate feedback into the those patterns as well as the GITTs developed for the effective adoption of the valuefied patterns. This can be achieved through:

- **(A4)** Gathering and analyzing insights on the value aspects of the design patterns.
- **(A5)** Collecting and analyzing feedback on the adoption of the valuefied design patterns.
- **(A6)** Maintaining an up-to-date list of the value breaches, valuefied design patterns, and GITTs.

2 BACKGROUND

Decades of software engineering research has led to a range of effective and successful software development methods. The history is such that early methods of Software Engineering focused on how to get the functionality correct, and subsequent methods focused on non-functional requirements (NFRs) such as performance, safety and security [3]. In the case of NFRs software methods evolved towards a recognition that each NFR needs to be explicitly managed as early in the software development lifecycle as possible, rather than being considered an afterthought. In the case of security, this mitigated the negative economic impact of security breaches in software systems [7]. Some of the noteworthy approaches to embed values in software design are Values Sensitive and Value Centered Design [10, 17], Participatory [28] and Values-Led participatory Design [21]. These approaches however, lack systemic thinking [12] and remain restricted to early stages of development. Despite the availability of several Software Engineering codes of ethics [19, 20, 30] and recent initiatives such as TechEthics conference [8] urging technologist to prioritize and adhere to relevant social values in technology creation, violations of values persist in various software domains such as Robotics, Machine Learning and AI [6], Software Design [35] and nanotechnologies [34].

In software engineering, a cross-cutting concern is a feature that cannot easily be isolated to a single module, but rather affects the entire application. Typical examples are logging and security. Social values are in fact good examples of cross-cutting concerns. When designing a workplace HR system to be sensitive to gender equality, for example, the designer will not be able to easily isolate all gender equality issues in one place, as gender considerations will be found in leave request modules, promotion modules, flexible working practice monitoring modules, etc. It is, however, important to have an overall view on the social value of gender equality; otherwise, some modules may not handle the value properly. This is where design methods for cross-cutting concerns, such as MATA (Modeling Aspects Using a Transformation Approach) [38], can help, as they allow the designer to switch easily back and forth between a value-summary view and a module-specific view.

Traditional 3GL programming languages were not able to isolate such cross-cutting examples in one place; this led to the development of aspect-oriented languages, such as AspectJ, that allow the designer to specify the cross-cutting design in one place and a compiler automatically distributes the concern across the entire application. MATA tool [38] was the first to provide a formal underpinning for handling cross-cutting concerns during software design; designers could specify design concerns as UML 'fragments' and a graph-theoretic transformation engine then 'weaved' these fragments into the main design. MATA also provided static analysis techniques to automatically detect conflicts between fragments – this is important because cross-cutting concerns typically conflict with each other (for example, the introduction of security mechanisms will usually lead to a downgrade in performance).

Whilst the software engineering community has focused heavily on safety, security and other critical NFRs in recent years, there has been comparatively little focus on broader social values – such as inclusion, integrity, well-being, transparency, to name a few [22]. Yet, it has been shown that a lack of consideration of social values in software systems can lead to the same consequences as a major security breach. In the case of the Volkswagen emissions scandal, software was deliberately designed in contradiction to the company's corporate values of "for responsible thinking", a decision that led to the resignation of the CEO, a 30% drop in VW's stock price and 25% drop in sales within a year. This brutal economic impact, felt more broadly across the entire automotive industry, was a direct result of a misalignment of the company's stated corporate values and the realization of its values in its software systems. This is not, however, an isolated example of where software inherently embodies values misaligned with those values which are claimed.

Galhotra et al. [18] cite many examples of 'biased 'software systems, which act in a way different to the values intended by its designers; supply and demand software pricing systems that unexpectedly led to price gouging on airline tickets for those trying to evacuate from Hurricane Irma – as the New York Times reported at the time, "There are no ethics valves built into the system that prevent an airline from overcharging during a hurricane."

In short, whether deliberately or unintentionally, there are now countless examples of where software has not been designed with social values in mind and, because of this, billions of dollars are routinely wiped off company market capitalizations or passed on to the consumer in terms of price hikes. This is in addition to the less tangible social impacts of such breaches. We argue that there is currently no software development methodology that can be used to properly align stated social values (as stated by an organization) and the implicit values that a software system embodies. Although there are existing guidelines for considering certain aspects of values and ethics (e.g., ACM code of ethics), current software methods do not provide a systematic way of specifying, designing and monitoring these values throughout a software development process.

3 THE VALUE-DESIGN HUB

To achieve aims (A1)-(A6) described in Section 1, we outline a semi-formal framework, referred to as *Value-Design Hub* (VDH), that lays a solid foundation for collaborative valuefication of design patterns. The framework will be developed in participation with software designers, users, and social scientists. VDH comprises of six components as depicted in Figure 1, where each component (C1)-(C6) corresponds to its respective aim in (A1)-(A6) :

(**C1**) VDH Classifier: includes a set of *Guidelines, Indicators, Tools, and Techniques* (GITTs) for the classification of the existing design patterns by specifying their value implications on software;
(**C2**) VDH Pattern Maker: includes a set of GITTs for extending and/or developing a set of design patterns that explicitly take into account social values;
(**C3**) VDH Guide: provides a set of GITTs that facilitate the adoption of the valuefied design patterns in software projects;
(**C4**) VDH Connector: includes a set of GITTs for gathering and analyzing the insights on the value aspects of the design patterns;
(**C5**) VDH Monitor: includes a set of GITTs for monitoring, collecting, and analyzing feedback on the adoption of the valuefied design patterns;
(**C6**) VDH Maintainer: keeps track of different variations of the valufied design patterns as well as GITTs provided by (C1)-(C5). VDH Maintainer further allows for updating design pattern and the GITTs provided by VDH Guide.

Components (C1)-(C3) are directly concerned with the valuefication of the design patterns while (C4) provides input to (C1)-(C3). Also (C5) enables monitoring the feedback on the adoption of the valuefied patterns. In other words, VDH Monitor helps enhance the effectiveness of VDH Classifier, VDH Pattern Maker, and VDH Guide by incorporating the feedback from collaborators. Moreover, (C6) maintains the valuefied design patterns as well as the GITTs provided by (C1)-(C5). The proposed VDH framework and its main components will be validated by studying real-world software projects.

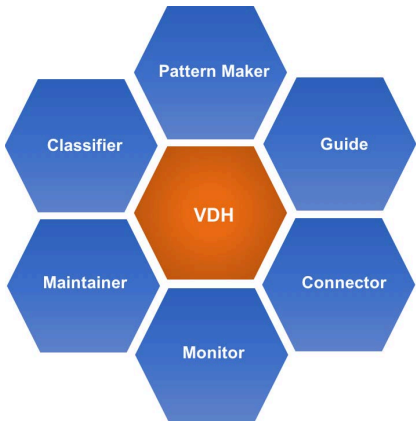

Figure 1: The Value-Design Hub and its Components.

Figure 1 demonstrates the major components of VDH at the highest level of abstraction. These components will be developed as the main outcomes of the research to integrate social values into software design in a collaborative manner. The main collaborators of VDH are users as well as the experts in software design and social sciences. They will collaborate over valuefication of the design patterns as depicted in Figure 2. The collaborates moreover, provide insights on the valuefication process. Valufication may be carried out in three different ways depending on the input/insights collected from collaborators. One is to classify an existing design pattern by specifying the value implications of the pattern on software. This can be achived using the Classifier component of VDH. Another way to valuefy design patterns is to extend an existing pattern or develop a new pattern using VDH Pattern Maker. Finally, the valuefication of design patterns can be assisted by providing guidance on the adoption of the valuefied design patterns, which is supported by VDH Guide. Figure 2 gives the use case diagram of VDH, where the main functionalities of VDH, corresponding to the VDH components, and their association with different types of collaborators are demonstrated. The VDH admin in Figure 2 specifies human actors that are responsible for the integrity of the process of collecting insights from collaborators as well as updating the pattern library. Pattern library of VDH includes the list of described design patterns including the the valuefied patterns and their meta-data. The main functionalities of VDH are described in the following subsections.

3.1 Collecting Insights from Collaborators

VDH connector collects and manages three different types of inputs/insights that can be provided by software users and experts, i.e. collaborators, to be used as the input for the valuefication process (Figure 3). The first type of the inputs/insights includes design practices that are not included in the established design patterns, yet they can be used to account for social values in software. Such practices can be identified, for instance, by studying the design practices that have resulted in a significant improvement in social aspects of real-world software. These practices are used to extend and/or develop design patterns that assist integrating social values into software. Moreover, the existing design patterns may positively or negatively influence social values in software products. In this regard, studying the value-related impacts of adopting different design patterns will help provide guidance on the value-conscious adoption of those patterns in software projects. The second type of the inputs/insights for VDH hence are required to help specify the value implications of the existing design patterns. Finally, the third type of the inputs/insights contains the knowledge on the effective use of the valuefied design patterns. This knowledge comes from the collaborators and will be made available to the designers in terms of *Guidelines, Indicators, Tools, and Techniques* (GITTs) that assist a value-conscious adoption of design patterns. To create an initial set of GITTs for VDH Connector, we are currently studying, in collaboration with social scientists and software practitioners, the implicit and embedded social values in existing software design patterns. We are further carrying out industrial studies to investigate the effectiveness of the existing design techniques in value-conscious software design.

3.2 Classification of the Design Patterns

The design patterns used in software projects convey different values, which may or may not be aligned with the values of the society for which that software product is designed. In other words, design patterns may have value implications, which in many cases

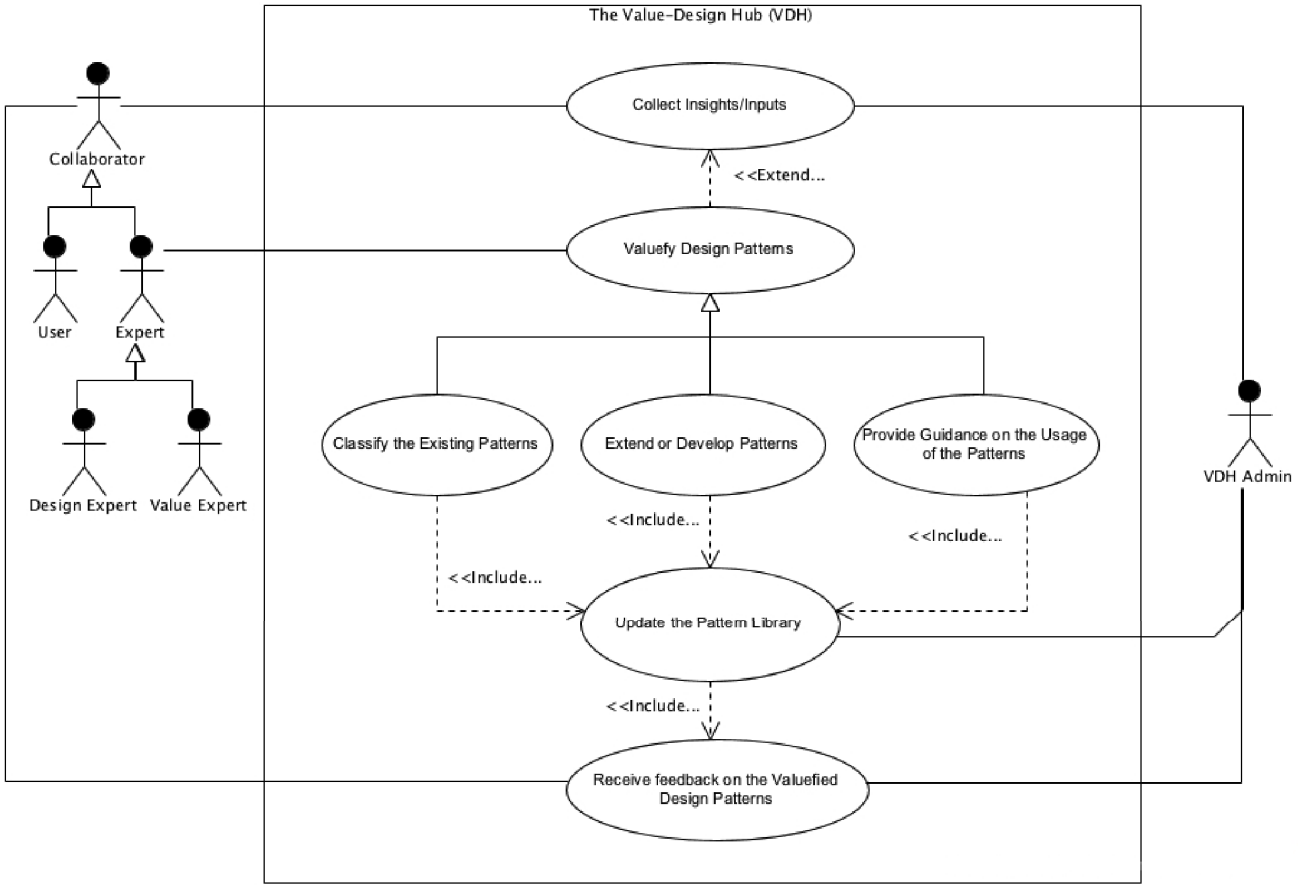

Figure 2: The use case diagram of the Value-Design Hub.

are not explicitly stated in the description of those patterns. One of the main aims of this research is to develop VDH Classifier to help investigate the value implications of the existing software design patterns based on the insights gathered through collaborating with the designers and users of software products as well as the experts in social sciences. This further helps understand and classify anti-patterns that are mainly unintended and caused by the indirect consequences of using certain design patterns, applying them in a wrong context, or using them for a wrong audience (individuals/society). Classified design patterns can be semi-formally described to enable automated analysis of those patterns. Moreover, graph-based techniques can be explored to enable analysis of relations and conflict among the value aspects of the design patterns. To build a foundation for VDH Classifier, we are currently studding the value implications of the existing software design patterns and anti-patterns in software projects. Our studies are designed in collaboration with social scientists and software practitioners from different areas including digital health and financial sector.

3.3 Extending or Developing Patterns

VDH Pattern Maker is aimed to valuefy design patterns by either extending the existing patterns in a way that they account for social values or developing new design patterns that integrate social values into software. In either case, design practices, which may not belong to the existing design patterns, but they properly address social aspects of software, can be described as valuefied design patterns and made available to software practitioners. To assist this, a semi-formal description language will be devised for the extension or creation of the design patterns. Such language will further enable semi-automated analysis of the design patterns. We are currently studying best practices in real-world software that assist value-conscious design of software products. Those practices will be used to propose an initial set of the extended or developed design patterns that integrate social values into software.

3.4 Providing Guidance on the Adoption

The usage of the design patterns includes selection and adoption of those patterns. Selection of the design patters is non-trivial due to the value dependencies [25–27] among different design patterns. In other words, a design pattern may help integrate a certain type of value into software while causing a breach in other value types. Moreover, the effectiveness of a valuefied design pattern may change in the presence or absence of other design pattens. These factors make selection of the design patterns a complex task, which

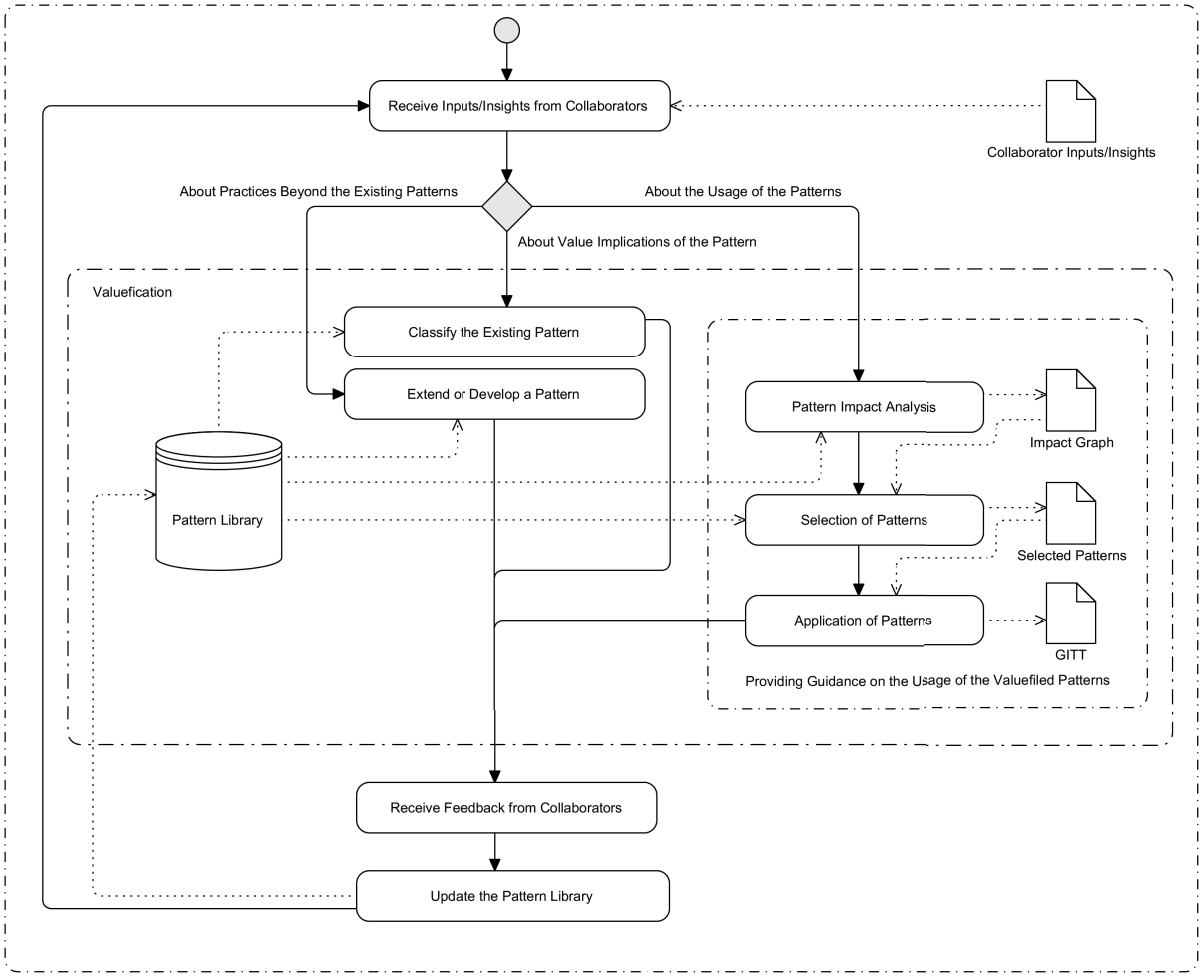

Figure 3: Process Flow in Value-Design Hub.

requires analyzing the positive and negative dependencies among the value implications of design patterns. To assist this, we develop a decision support system, which allows for analyzing the impacts of different design patterns on the values the society in which a software product is intended to be used. Moreover, the gap between the design and implementation has always been a major challenge in software development [29], which also affects the adoption of design patterns. To mitigate this we develop a set of GITT that facilitate a proper adoption of the design patterns.

3.5 Monitoring the Adoption of the Patterns

Valuefication of software design patterns in VDH will be achieved by VDH Classifier, VDH Pattern Maker, and VDH Guide. VDH Connector connects the users, software designers, and social scientists to collaboratively valuefy design patterns. Feedback on real-world adoption of the valuefied design patterns and the GITTs provided by VDH framework is also very important. Collecting such feedback, analyzing them, and incorporating them into VDH is essential to the collaborative and evolutionary nature of the framework. This helps iteratively enhance the effectiveness of VDH in integrating social values into software design. In this regard, establishing a social monitoring system for incorporating real-time social media data will help power VDH Monitor. VDH will also include techniques for identifying value breaches in software design patterns and/or the use of those patterns by monitoring users behavior such as their usage pattern, facial expressions, and several other indicators, which can be monitored using machine learning.

3.6 Maintaining the Valuefied Patterns

Maintaining an up-to-date library of the valuefied design patterns helps designer to systematically get access to the best practices that assist a value-conscious software design and avoid using patterns that may result in breaching social values in software. Moreover, keeping a list of the value breaches will assist software designers in devising patterns that mitigate such breaches in software. The necessity of this has been observed by the projects such as NVD (National Vulnerability Database) [39] in which security aspects are taken into account. VDH on the other hand, keeps record of the

design patterns and value breaches concerning several aspects of social values including security, conformity, tradition, benevolence, self direction, and so on. Hence, one of the main outcomes of VDH as depicted in Figure 3 is a library of the valuefied design patterns, which includes a list of semi-formally described design patterns and their corresponding met-data. Moreover, VDH components include/provide different GITTs for the integration of social values into software design. Such GITTs are subject to frequent changes based on the insights/feedback provided by the users as well as the experts in software design and social sciences. The VDH framework further provides knowledge sharing mechanisms such as forums and wikis to enable collaborations among software users, designers, and social scientists on the integration of social values into software. For such wide scale collaborations, among VDH components as well as the users and experts, effective techniques are required to facilitate maintaining an up-to-date list of the design patterns, GITTs, and all the data gathered from collaborates. This will be managed by VDH Monitor. Finally, VDH Monitor can customize the GITTs depending on the values of the society for which a software product is designed.

3.7 A Valuefication Example

User interface design patterns (UIDPs) as a subcategory of SDPs are widely used in software projects [24, 37]. Social values however, may impact the design of the user interface of an application or website [37]. These values manifest themselves in terms of choices of symbols, heroes/heroines, and rituals, which vary across different cultures. To reduce the risk of rejecting software thus, it is important to account for the values of the target society in UIDPs. One of the interesting examples of how value differences impact the design of software user interfaces is discussed by Marcus et al. [24]. In their work Marcus et al., highlighted the differences between the *Individualism* in Western societies (specifically USA) and *Collectivism* in the Costa Rican culture. The authors observed certain features in the user interface design of the websites of two different national parks, one from the USA and the other from Costa Rica, manifesting this cultural difference. Marcus et al. [24] explained that the emphasis in the US website was more on the aims of the visitor and possible actions in coming to the park. The Costa Rican website on the other hand, featured an emphasis on the nature, downplayed the individual tourist, and used a slogan to promote a national agenda. An even more startling difference was that instead of a typical Western display of consuming new technologies or experiences, the website was filled with several political announcement that the Costa Rican government has signed an international agreement against the exploitation of children and adolescents.

One way to valuefy design patterns in VDH is to classify those patterns based on their value implication. In the case of the national park example, the UIDP used for designing the website of the US national park can be classified by the Classifier component of VDH as *Individualist*, whereas the UIDP used for the Costa Rican website can be classified as *Collectivist*. This helps designers of new websites to choose a user interface design pattern[37] that suits the value of their end user whether that is to be more *Individualist* or *Collectivist*. To facilitate this, the Maintainer component of VDH keeps record of an up-to-date library of the semi-formally described valuefied design patterns with their corresponding meta-data. It is worth mentioning that a valuefied design pattern may, simultaneously, belong to different classes of values. For instance, a UIDP classified as *Individualist* may also emphasize on *Femininity* by using features that promote modesty, tenderness, and a concern with both quality of life and material success [24]. Information that help valuefy UIDPs may come from users, designers, or social scientists that examine the websites/applications with respect to their value implications. This is facilitated by the Connector component of VDH. In the case of the national park example, examiners of the websites observed features that finally classified them as *Individualist* or *Collectivist*. Now the question is do users of the website feel the same way? this is specially important when the website is intended to be *Individualist* or *Collectivist*. To find this out, the Monitor component of VDH helps receive feedback on the effectiveness of the valuefied design patterns in embedding the intended social values into software. Such feedback can be used to further improve the effectiveness of the valuefication in VDH.

4 CONCLUSIONS

In this paper we discussed the adverse impacts of breaching social values in software design. We further elaborated and provided examples, such as the Hurricane Irma, on how ignoring social values may result in breaching social values and ultimately lead to user dissatisfaction, lack of proper adoption, financial loss, and other social implications. To mitigate the adverse impacts of breaching social values in software, this paper proposed a framework for collaborative integration of social values into software design patterns. Through collaborations with users, designers, and social scientists, the proposed framework, referred to as value-design hub (VDH), allows for (i) specifying the value implications of the design patterns, (ii) extending or developing design patterns that specifically account for social values, (iii) providing guidance on the adoption of the design patterns with respect to social values, (iv) collecting and analyzing insights from collaborates, (v) maintaining an up-to-date library of the valuefied design patterns, and (vi) collecting collaborators' feedback and incorporating those feedback into the framework.

REFERENCES

[1] Brian Benchoff. 2015. Ethics in engineering: Volkswagen's diesel fiasco. http://hackaday.com/2015/09/23/ethics-in-engineering-volkswagens-diesel-fiasco

[2] Alofi Shane Black and Tony Sahama. 2016. Chronicling the patient journey: co-creating value with digital health ecosystems. In *Proceedings of the Australasian Computer Science Week Multiconference*. ACM, 60.

[3] Barry Boehm. 2006. A view of 20th and 21st century software engineering. In *Proceedings of the 28th international conference on Software engineering*. ACM, 12–29.

[4] Grady Booch. 2014. The Human and Ethical Aspects of Big Data. *IEEE Software* 31, 1 (jan 2014), 20–22. https://doi.org/10.1109/MS.2014.16

[5] Geoffrey Bowker, Susan Leigh Star, Les Gasser, and William Turner. 2014. *Social science, technical systems, and cooperative work: Beyond the great divide*. Psychology Press.

[6] Joanna Bryson and Alan Winfield. 2017. Standardizing Ethical Design for Artificial Intelligence and Autonomous Systems. *Computer* 50, 5 (may 2017), 116–119. https://doi.org/10.1109/MC.2017.154

[7] Huseyin Cavusoglu, Birendra Mishra, Srinivasan Raghunathan, and Published M E Sharpe. 2014. The Effect of Internet Security Breach Announcements on Market Value : Capital Market Reactions for Breached Firms and Internet Security Developers Source : International Journal of Electronic Commerce , Vol . 9 , No . 1 , Measuring the Business Value of . 9, 1 (2014), 69–104.

[8] Raja Chatila, Kay Firth-Butterflied, John C. Havens, and Konstantinos Karachalios. 2017. The IEEE Global Initiative for Ethical Considerations in Artificial Intelligence and Autonomous Systems [Standards]. *IEEE Robotics & Automation Magazine* 24, 1 (mar 2017), 110–110. https://doi.org/10.1109/MRA.2017.2670225

[9] Tony Clear. 2013. Novopay: dilemmas in a nearshore outsourcing project failure [invited presentation]. (2013).

[10] Gilbert Cockton. 2005. A development framework for value-centred design. In *CHI'05 extended abstracts on Human factors in computing systems*. ACM, 1292–1295.

[11] Anita H M Cremers, Mark A Neerincx, and Jacomien G M De Jong. 2013. Inclusive design: Bridging theory and practice. *10th International Conference on Engineering Psychology and Cognitive Ergonomics: Applications and Services, EPCE 2013, Held as Part of 15th International Conference on Human-Computer Interaction, HCI 2013, 21 July 2013 through 26 July 2013, Las Vegas, NV* (2013), 323–332. http://dx.doi.org/10.1007/978-3-642-39354-9-35{%}5Cnhttp://xref.tno.nl/bibliotheek/sv-015068/TNO/Publicaties/2013/cremers-2013-inclusive.pdf

[12] Maria Angela Ferrario, Will Simm, Jon Whittle, Christopher Frauenberger, Geraldine Fitzpatrick, and Peter Purgathofer. 2017. Values in Computing. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '17*. ACM Press, New York, New York, USA, 660–667. https://doi.org/10.1145/3027063.3027067

[13] Eduardo Figueiredo, Bruno Silva, Claudio Sant'Anna, Alessandro Garcia, Jon Whittle, and Daltro Nunes. 2009. Crosscutting patterns and design stability: An exploratory analysis. In *Program Comprehension, 2009. ICPC'09. IEEE 17th International Conference on*. IEEE, 138–147.

[14] Andrew J Flanagin, Craig Flanagin, and Jon Flanagin. 2010. Technical code and the social construction of the internet. *New Media & Society* 12, 2 (2010), 179–196.

[15] Batya Friedman. 1997. *Human values and the design of computer technology*. Number 72. Cambridge University Press.

[16] Batya Friedman, Peter H. Kahn, Alan Borning, and Alina Huldtgren. 2013. Value Sensitive Design and Information Systems. Springer, Dordrecht, 55–95. https://doi.org/10.1007/978-94-007-7844-3_4

[17] Batya Friedman, Peter H Kahn, Alan Borning, and Alina Huldtgren. 2013. Value sensitive design and information systems. In *Early engagement and new technologies: Opening up the laboratory*. Springer, 55–95.

[18] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness testing: testing software for discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2017*. ACM Press, New York, New York, USA, 498–510. https://doi.org/10.1145/3106237.3106277

[19] Ph Galiay. 2009. A Code of Conduct for Responsible Nanosciences and Nanotechnologies Research in Europe. *Nanotec* 2009 (2009), 23.

[20] Donald Gotterbarn and Keith W. Miller. 2009. The Public is the Priority: Making Decisions Using the Software Engineering Code of Ethics. *Computer* 42, 6 (jun 2009), 66–73. https://doi.org/10.1109/MC.2009.204

[21] Ole Sejer Iversen and Tuck W. Leong. 2012. Values-led participatory design. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction Making Sense Through Design - NordiCHI '12*. ACM Press, New York, New York, USA, 468. https://doi.org/10.1145/2399016.2399087

[22] Deborah G Johnson and Helen Nissenbaum. 1995. *Computers, ethics & social values*. Prentice-Hall, Inc.

[23] Deborah G. Johnson and Helen Fay. Nissenbaum. 1995. *Computers, ethics & social values*. Prentice Hall. 714 pages. https://dl.acm.org/citation.cfm?id=206759

[24] Aaron Marcus and Emilie West Gould. 2000. Crosscurrents: cultural dimensions and global Web user-interface design. *interactions* 7, 4 (2000), 32–46.

[25] Davoud Mougouei. 2016. Factoring requirement dependencies in software requirement selection using graphs and integer programming. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering - ASE 2016*. ACM Press, New York, New York, USA, 884–887. https://doi.org/10.1145/2970276.2975936

[26] Davoud Mougouei, David M. W. Powers, and Asghar Moeini. 2017. An Integer Linear Programming Model for Binary Knapsack Problem with Dependent Item Values. Springer, Cham, 144–154. https://doi.org/10.1007/978-3-319-63004-5_12

[27] Davoud Mougouei, David M. W. Powers, and Asghar Moeini. 2017. Dependency-aware software release planning. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 198–200. https://doi.org/10.1109/ICSE-C.2017.74

[28] Michael J Muller and Sarah Kuhn. 1993. Participatory design. *Commun. ACM* 36, 6 (1993), 24–28.

[29] Gail C Murphy, David Notkin, and Kevin J. Sullivan. 2001. Software reflexion models: Bridging the gap between design and implementation. *IEEE Transactions on Software Engineering* 27, 4 (2001), 364–380.

[30] Michael D Myers and John R Venable. 2014. A set of ethical principles for design science research in information systems. *Information & Management* 51, 6 (2014), 801–809.

[31] Queensland. Department of Justice, Attorney-General, and Richard Chesterman. 2013. *Queensland Health Payroll System Commission of Inquiry: Report*. Queensland Department of Justice and Attorney-General.

[32] Bryan Pfaffenberger. 2017. Technological Dramas Author (s): Bryan Pfaffenberger Published by : Sage Publications , Inc . Stable URL : http://www.jstor.org/stable/690096 Technological Dramas. 17, 3 (2017), 282–312.

[33] Alina Pommeranz, Christian Detweiler, Pascal Wiggers, and Catholijn M Jonker. 2011. Self-reflection on personal values to support value-sensitive design. In *Proceedings of the 25th BCS Conference on Human-Computer Interaction*. British Computer Society, 491–496.

[34] Arie Rip and Douglas K. R. Robinson. 2013. Constructive Technology Assessment and the Methodology of Insertion. 37–53. https://doi.org/10.1007/978-94-007-7844-3_3

[35] Marie Louise Juul Søndergaard and Lone Koefoed Hansen. 2017. Designing with Bias and Privilege?. In *Nordes 2017*, Vol. 7.

[36] RN Taylor and A Van der Hoek. 2007. Software design and architecture the once and future focus of software engineering. *Future of Software Engineering, …* (2007). http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=4221623

[37] Jenifer Tidwell. 2010. *Designing interfaces: Patterns for effective interaction design*. " O'Reilly Media, Inc.".

[38] Jon Whittle and Praveen Jayaraman. 2007. MATA: A Tool for Aspect-Oriented Modeling Based on Graph Transformation. In *Models in Software Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg, 16–27. https://doi.org/10.1007/978-3-540-69073-3_3

[39] Su Zhang, Doina Caragea, and Xinming Ou. 2011. An empirical study on using the national vulnerability database to predict software vulnerabilities. In *International Conference on Database and Expert Systems Applications*. Springer, 217–231.