An *Approximate* Theory for Value Sensitivity

Balbir Barn*, Ravinder Barn†
*School of Science and Technology, Middlesex University, London
{b.barn}@mdx.ac.uk
†Royal Holloway University of London
{r.barn}@rhul.ac.uk

***Abstract*—The software engineering community is on a quest for general and specific theories for the discipline. Increasingly, systems constructed for today's hyper connected world are raising issues of security and privacy, both examples of value concerns. Hence there is a need to articulate a theory for value sensitivity that software engineers can draw upon to evaluate their designs and to embed outcomes into systems that are developed. This paper proposes an *approximate* theory of value sensitivity in recognition that this is a journey and an interim struggle. The theory is articulated using the framework proposed by Sjøberg et al. An initial evaluation is provided for both the value sensitivity theory and the framework.**

***Index Terms*—Value Sensitive Design, Values, Co-Design, theory building, software engineering.**

I. Introduction

There is an ongoing quest to establish both a general theory of software engineering [http://semat.org] and a better understanding of relevant specific theories that can be applied to the software engineering domain. Recall that Software Engineering (SE) as defined by Somerville is: "An engineering discipline which is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use." [14]

By virtue of its Wittgensteinan family resemblance to engineering more broadly [19], SE applies theories, methods and tools in a selective manner. Further more, the broadness and scope of activities also requires SE to be practiced within organisational and human based interaction constraints.

One issue with respect to both a general theory and the use of specific theories is that software engineering is both a domain in itself as well as its practice being applied in a variety of domains. Hence the range and applicability of a particular theory is particularly challenging.

Software engineering is also tricky with respect to theories because it solves problems in a societal context. Hence SE practice should really be considered a socio-technical activity. Yet our literature and practice is relatively limited with respect to the interaction between SE and socio-technical systems. Of obvious interest in this paper is the issue of how values are addressed in SE and in particular, the issues of ethical concerns.

The word "theory" suffers from both an over-use and a reluctance in its use by researchers. Weick comments that most theories that are labeled as theories are actually *approximate* theory in that they go some way to establishing a theory but fall short in some aspect such as: failing to sufficiently articulate relationships between variables/concepts contributing to the theory; or perhaps making post-factum interpretations whereby ad hoc hypotheses are derived from limited observations [17]. The reluctance to use the word "theory" is illustrated most aptly by Runkel and Runkel [12] reported in [17] where titles such as *Approach to a Theory o*f . . , *Notes Toward a Theory of* ... are frequent.

This strikes a chord. In this paper, as our title indicates, we are proposing a theory, moreover the proposed theory is along the continuum of theory development or "interim struggle", and is therefore an *"approximate"* theory.

A. Values, the need for theory and this contribution

Software engineering is integral to societal sustainability in that most sustainability issues require inter-connection and human interactions with systems. Thus research in socio-technical systems is needed to encompass society, organisations and individuals and their behaviour [11]. A basic driver of human behaviour may at least in part be explained by a notion of "Value" [10]. Hence, a discussion about how values can be exposed, discussed, mediated and integrated into the design of socio-technical systems is necessary.

In this paper we hypothesise that part of the problem of systems design is that current SE practice (in any of its variants) does not incorporate any theoretical perspective of value as a first class representation that can support such a discussion. A simple test of searching for the term "value" in one of the leading texts on software engineering (Somerville, 6th Edition [14]) demonstrates its absence.

Thus, this paper contributes an *approximate* theory of value sensitivity that can be used in software engineering practice to address the socio-technical aspects of modern information systems. By drawing on the constructs of Sjøberg et al [13], we provide a semi-formal representation of the theory and in using these constructs, we provide a critique of the both value sensitivity theory and technology of Sjøberg et al.

The remainder of this paper introduced the basic artefacts for theory description and construction (Section 2). Concepts and existing work in value sensitivity presented in section 3. Section 4 presents the basic components of the theory. Section 5 provides some discussion and concluding remarks.

II. Theory construction and description

Space does not allow for a detailed philosophical discussion of the nature of a theory, instead, we draw on some representat-

Archetype class	Description
Actor	a role played by a user or any other system that interact with the subject
Technology	any process, model, method, technique, tool or language construct.
Activity	a piece of behaviour at any level of granularity
Software System	a general denotion of a system that can be classified in many dimensions such as size, domain, administrative, distributed, embedded. etc.

Table I
TYPOLOGY FOR SE THEORIES (FROM SJØBERG ET AL [13]

ive descriptions to outline what is generally agreed, constitutes a theory [13], [18], [15]. The main elements of a theory are:

constructs:	the basic conceptual elements extracted from the domain of discourse that are generally measurable.
relations:	relations describe connections among constructs and their interactions with one another.
boundary:	a boundary of a theory describes its scope or the validity of the theory under certain conditions.
propositions:	statements that are concerned with making predictions about a theory's constructs.

A theory for use in SE should therefore be described in these terms. But the peculiarity of SE demands that a theory both meets the needs of the domain of SE and indirectly other domains where software applications incorporating theories are developed. Hence, a somewhat meta discussion ensues whereby a framework, or rather an approximate theory is required to relate the theory under development to SE practice at large. We use the constructs outlined by Sjøberg et al. to present our approximate theory [13].

For a theory to be relevant to SE, it must explain or predict phenomena occurring in SE. In a critical sentence, Sjøberg et al. succinctly capture how to relate a theory in development to SE practice:

> "The typical SE situation is that an *actor* applies *technologies* to perform certain *activities* on an (existing or planned) *software system*. "[13, :p 10]

In this statement, they introduce what they call "archetype classes" and propose that for a theory *T* under development, theory elements of *T* should be typically associated with these archetype classes. Such a typology is a particular form of theory building in its own right [5]. These classes and tentative definitions are shown in table 1. The typology is supported by graphical notation based on the Unified Modelling Language (see example description of UML here [6]). When using this approach, Sjøberg et al. identify theory elements similar to that listed above.

In this paper, we will use both the typology and the UML notation to present the value sensitivity theory. The next section will first introduce the domain background for why this theory is relevant to SE practice.

III. VALUE SENSITIVITY

The background to value sensitivity or value sensitive design has been described elsewhere (see [1], [2]). Nonetheless it is useful to present a short summary here.

The foundations for discussing how values could be integrated into the SE process has largely been initiated through the domain of human-computer interaction (HCI) in the seminal article by Suchman [16] and has influenced the advent of Co-Design, where questions about values "become easier since the collaboration between designer and client (or user) is explicitly recognised as a goal of the process."[9].

A notable and sustained contribution to understanding and accounting for values in the design process is the work by Friedman and her colleagues on Value Sensitive Design (VSD)[7], [8]. Their research has informed the debate on the unintended consequences of systems and the detrimental effect or compromise of moral values that users may believe in.

Friedman defines value as: *"what a person or group of people consider important in life*". Values that are particularly pertinent to information systems include: ownership and property; privacy, freedom from bias, universal usability, trust, autonomy, informed consent, identity and others. In systems design, *values* have, to-date, been integrated mostly with participatory design approaches [3] or more recently Co-Design. Co-Design involves potential (un-trained) end users working jointly with researchers and designers to jointly create artefacts that lead directly to the end product and, as Yoo et al. [20] note: "become a dominant user study methodology in the fields of product design, service design, interaction design and HCI". Several researchers have commented that whilst participatory design is a dominant mode of technological design, end-users still struggle to influence the direction of design within the participatory process. Furthermore, end-users may not fully understand the ecology of available technologies. It may be that the reductionist principles of SE could be argued to have hindered the integration of values approaches into mainstream practice and so making it harder to monitor such concerns.

VSD emerged to integrate moral values (and more broadly ethics) with the design of systems. A key premise of VSD is that it seeks to design technology that accounts for human values throughout the design process (over and beyond the identification of functionality and visual appearance) of systems.

In the Yoo et al. model, the traditional co-design core blends methods from value-sensitive design to structure the co-design engagement with inputs from stakeholders and considerations of values. The co-design process may be initiated by free-form thinking, but their key innovation is in the introduction of two types of structured interventions. Designer prompts entail materials that originate from expert designers and may comprise personas, scenarios or the use of envisioning cards. Stakeholder prompts originate from the end-users and may utilise value based scenarios addressing concerns such as unintended uses of the system; changes of the use of the system over time and so on. Values will be, typically, those derived from the list suggested by Friedman in [7]. The reflection element of the model provides a way of representing how prompts may be generated by either a stakeholder or a designer as a result of joint participation in the co-design space.

IV. An approximate theory for value sensitivity

Currently there are no formal or semi-formal models and their associated processes for integrating values and requirements into the SE design process. Building on the work of Yoo et al. [20], we have developed a Co-design workshop method that exposes value concerns such as security and privacy. The approach has been applied on a research project (Mobile Apps for Youth Offending Teams – MAYOT) aimed at developing social technology for use by young people and their caseworkers in youth offending teams in the UK. The project raised requirements on design methods to incorporate the voice of stakeholders with respect to privacy and other moral value issues.

The Co-design activities in the various workshops yielded a rich set of data including design and specification of features/functions of the MAYOT app. We focus on one design feature. The *Exclusion Zone* feature is a function that is available on the MAYOT app that allows a case worker to define a geographic region from which a young person is prohibited (with potential risks to violating their youth order with obvious detrimental effects). The feature alerts the young person in possession of the smart phone hosting the app that they are in an exclusion zone. During the course of a number of workshops, the feature went through a number of design changes representing the designer's view, a case worker's view, and views from young people who believed that the prototype app feature violated their privacy. Such concerns were considered seriously by the research team to ensure that a balance was struck between relative privacy and security of information. This consideration creates rich descriptions of the requirements process that have hitherto not been captured. The essential purpose of our theory is to provide a mechanism for explaining these requirements discussions.

In presentation of the value sensitivity theory, constructs are represented as a UML class or attribute of a class. The archetype classes from Sjøberg et al. are shown as meta classes using <<stereotype>> notation. Constructs from the VS theory are shown as instances of meta types from the archetype classes. We contend that is a better way of providing a relationship between the described theory and their SE framework as it is more parsimonious in its use of space. Additional semantics (meta model versus model) also become available. Whilst there are other approaches for developing concpetual models, the use of UML allows the deployment of existing experience and knowledge of modelling and other features of UML to embellish descriptions of theories. The UML conceptual model is augmented by proposition relationship stereotypes that highlight possible propositions that can be used to test the theory. These are shown as directed arrow lines between classes. Additional sub-typing is also possible. The

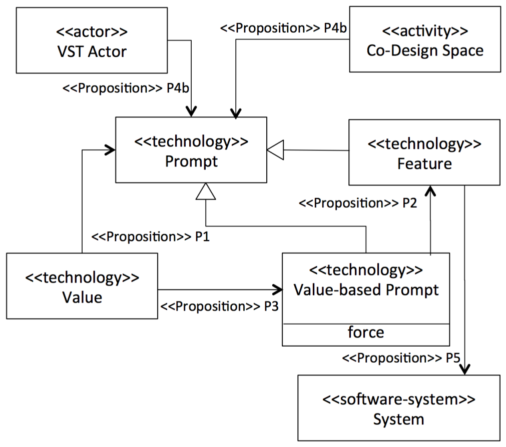

Figure 1. Value sensitivity theory

summary UML diagram (mostly conforming to the notation from Sjøberg et al) is shown in Figure 1.

Using the theory elements framework discussed in section 2. The following serves to describe the value sensitivity theory. The theory has been induced from our example project and design features but we suggest that it is a general theory that can be applied in other contexts.

Constructs:

VSTActor	A *VSTActor* is a designer, or stakeholder who participates in joint actions to create or evolve design features of intended systems. *VSTActors* are the source of the generation of *Prompts.*
Co-Design-Space	is any space where key VSTAc*tors*, the *Designers* and *Stakeholders* (including indirect stakeholders, elided in the diagram) participate in joint actions to create or evolve design *Features* of intended systems. In our example they are workshops.
Prompt	is a mechanism or prop that provokes reflection in the *Co-Design Space.*
Value-based-Prompt	is a *prompt* that raises concerns or addresses a *value* of interest to a *VSTActor*.
Value	is what a person or group of people consider important in life.
Feature	is a *Prompt* that has been accepted as an intended function of the socio-technical *System* as a result of discussion at least one or more co-design spaces.
Force	Each *VSTActor* will experience a different interpretation of value embedded in a *Value Based Prompt*. This interpretation is captured by the *Force* concept, represented as a numeric value for simplicity. A *Force* must be balanced between competing actors before a *Value Based Prompt* can be accepted as a *Feature*.

Propositions

These are statements that are concerned with making predictions about a theory's constructs. The propositions define relationships between constructs as shown in figure 1. The following propositions have been identified.

P1:	The ability to describe how *Values* influence *Prompts* leads to better socio-technical systems design.
P2:	The ability to describe how *Value Based Prompts* become *Features* leads to better socio-technical systems design.
P3:	*Value Based Prompts* can only become *Features* when associated *Forces* are balanced.
P4a,P4b:	The ability to describe how *VSTActors* generate prompts in a *Co-Design Space* leads to better socio-technical systems design and overall acceptance of a *System*.
P5:	*Features* that have been subject to value sensitivity leads to socio-technical *systems* that respect *Values* that *VSTActors* see as important.

Finally, the theory is bounded or scoped in limit as it is relevant to the SE of socio-technical systems. I.e. those that raise issues of values in their daily use. The mobile app developed as part of the MAYOT project is one such system.

V. Discussion and concluding remarks

There are two theories under discussion here. The first is the use of the typology presented by Sjøberg et al. Some initial comments can be made. The typology makes it possible to relate T constructs with SE practice. To our knowledge, there have only been limited independent efforts to apply this typology to an SE theory so there is a lack of empirical support. One such example is work from Santos et al. [4]. This lacunae is curious and suggests that there may be limitations in the underlying meta concepts such as omission or ambiguity. We have used the *Technology* concept to represent *Value* and *Prompts*. This seems to overload the meaning of Technology and perhaps there is a missing meta concept. The UML diagram extension and the notation (such as stereotypes) provides a visual and semi-formal representation of a theory's constructs and propositions. The use of meta classes (by way of stereotypes) provides an economical means of representation which we propose is an improvement on the use of inheritance in their original approach. While the typology is specific to SE practice, there is scope for a general purpose meta theory modelling language which could use this framework. Hence there are potentially multiple meta levels.

As we have argued initially, the Value sensitivity theory is approximate. The constructs and the general motivation for this theory arose through a process of induction from an ongoing research project and currently there is limited empirical evaluation of this approach. In contrast, though, there is ample qualitative evidence of value sensitivity concerns and how they might affect acceptance of a system. A small set of constructs are provided and so help in reducing the cognitive burden. Potential for widespread utility for the theory is significant given the increasing concern on how privacy, a core moral value, is being eroded by socio-technical systems delivered through apps on smart phones. Even within the limits of the experiment described here, the typology and its instantiation in the case of VST, suggests that a richer conceptual theory building language is required to fully address the constraints implicit in the proposed theory.

Acknowledgements

This research was partially funded by the Nominet Trust. We thank Franco Raimondi and the anonymous reviewers who provided many useful comments.

References

[1] Balbir S. Barn, Ravinder Barn, and Giuseppe Primiero. The role of resilience and value for effective co-design of information systems. In *Proceedings of the Conference of the International Association for Computing and Philosophy (IACAP 14)*. To appear in: Synthese Library, Springer, 2015.

[2] Balbir S. Barn, Ravinder Barn, and Franco Raimondi. On the role of value sensitive concerns in software engineering practice. In *36th International Conference on Software Engineering, ICSE Companion*, (Accepted) 2015.

[3] G. Bjerknes, P. Ehn, M. Kyng, and K. Nygaard. *Computers and democracy: A Scandinavian challenge*. Gower Pub Co, 1987.

[4] Paulo Sérgio Medeiros dos Santos and Guilherme Horta Travassos. On the representation and aggregation of evidence in software engineering: A theory and belief-based perspective. *Electronic Notes in Theoretical Computer Science*, 292:95–118, 2013.

[5] D Harold Doty and William H Glick. Typologies as a unique form of theory building: Toward improved understanding and modeling. *Academy of Management Review*, 19(2):230–251, 1994.

[6] Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Languange*. Addison-Wesley Professional, 2004.

[7] Batya Friedman. Value-sensitive design. *Interactions*, 3(6):16–23, 1996.

[8] Batya Friedman and Helen Nissenbaum. Bias in computer systems. *ACM Transactions on Information Systems (TOIS)*, 14(3):330–347, 1996.

[9] Christopher A Le Dantec and Ellen Yi-Luen Do. The mechanisms of value transfer in design meetings. *Design Studies*, 30(2):119–137, 2009.

[10] Edwin A Locke. The motivation sequence, the motivation hub, and the motivation core. *Organizational behavior and human decision processes*, 50(2):288–299, 1991.

[11] Lynette I Millett, Deborah L Estrin, et al. *Computing Research for Sustainability*. National Academies Press, 2012.

[12] Philip Julian Runkel and Margaret Runkel. *A guide to usage for writers and students in the social sciences*. Number 382. Rowman & Littlefield, 1984.

[13] Dag IK Sjøberg, Tore Dybå, Bente CD Anda, and Jo E Hannay. Building theories in software engineering. In *Guide to advanced empirical software engineering*, pages 312–336. Springer, 2008.

[14] Ian Sommerville. Software engineering. international computer science series, 2004.

[15] Klaas-Jan Stol and Brian Fitzgerald. Uncovering theories in software engineering. In *Software Engineering (GTSE), 2013 2nd SEMAT Workshop on a General Theory of*, pages 5–14. IEEE, 2013.

[16] Lucy Suchman. Do categories have politics? the language/action perspective reconsidered. In *Human values and the design of computer technology*, pages 91–106. Center for the Study of Language and Information, 1997.

[17] Karl E Weick. What theory is not, theorizing is. *Administrative Science Quarterly*, pages 385–390, 1995.

[18] Roel Wieringa, Maya Daneva, and Nelly Condori-Fernandez. The structure of design theories, and an analysis of their use in software engineering experiments. In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pages 295–304. IEEE, 2011.

[19] Ludwig Wittgenstein. *Philosophical investigations*. John Wiley & Sons, 2010.

[20] Daisy Yoo, Alina Huldtgren, Jill Palzkill Woelfer, David G Hendry, and Batya Friedman. A value sensitive action-reflection model: evolving a co-design space with stakeholder and designer prompts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 419–428. ACM, 2013.