Social Responsibility in System Design
(Extended Abstract)*

Huib Aldewereld and Virginia Dignum

Delft University of Technology, The Netherlands
e-mail:{h.m.aldewereld, m.v.dignum}@tudelft.nl

1 Introduction

During the design of a new technology (and, subsequently, software) many design choices need to be made at a high level of detail. These detailed design decisions shape the nature of the resulting software. The reasoning underlying each decision is ultimately based on abstract social and organisational values like, e.g., integrity, trust, security or fairness. These values are, however, typically only implicitly involved in the design process. As a result, the software might exhibit different characteristics than were foreseen because it is not *in compliance with* these abstract values. To ensure compliance, values should be systematically incorporated in the design. We propose a Value Sensitive Design approach to incorporating social responsibility in software.

The problem in current software development approaches is that values are only *implicitly* involved in the development process. The link between the values and the application is, at best, implicit in the decisions made and the choices taken. In the requirements elicitation process [3], the values are translated to concrete requirements. These requirements are then used throughout the development process, and the related value is lost. However, due to their abstract nature, values can be translated to design requirements in more than one way. If the values and their translation to requirements are left implicit in the development process, one loses the flexibility of using alternative translations of those values.

In this paper, we propose a first framework for describing the links between values, requirements and software design. Making these links *explicit* allows for improvements in the traceability of (the effects of) the values throughout the development process. This traceability greatly increases the maintainability of the application. That is, if a value is to be implemented differently, the explicit links between the values and the application make it much easier to determine which parts of the application should be updated. We illustrate the use of the framework with an use-case example.

2 Value Sensitive Design

The recognition that values have an impact on the development of technology comes from the field of philosophy (ethics, in particular). Especially the topic of Value Sensitive Design (VSD) has seen much development recently. While VSD is applicable to

* This paper is an abstract of the work presented in [1].

many engineering disciplines [4], in the area of ICT it has mainly been applied to the analysis of the reasons for resisting the introduction of ICT solutions in organisations or society [5]. Such resistance can be the result of not taking into account potential conflicts that the solution might have with human values (e.g., security vs. privacy). To avoid such failures, investigations have been done on the relationship between specific values and technology (e.g., see [2]). Unfortunately, VSD has so far remained at a philosophical level: focussing on analyses and lacking formalisation of the relations between the values and the technology.

In [1] we propose the Value Sensitive Software Development (VSSD) framework (see figure 1) to integrate the principles of Value Sensitive Design with Service Oriented Architectures (SOA) into the design of software systems. The framework aims to bridge the gap between abstract values and concrete systems. VSSD consists of three views described at three different abstraction levels. First, the *Value View* describes the organisational values, norms, rules and procedures. This view largely overlaps with the values hierarchy concepts of [4]. Secondly, the *Domain View* models the context of the system based on domain specific elements, for example, existing system blueprints. Finally, the *Modelling View* defines the system's architecture based on the values and the domain characteristics. This view builds heavily on existing theory from SOA-based computing (see, e.g., [6]).

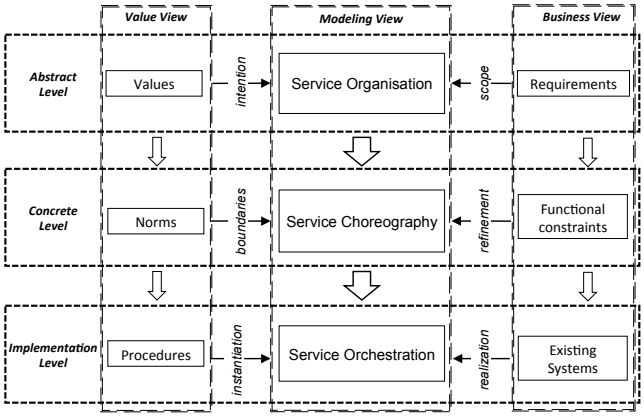

Fig. 1. Value Sensitive Software Development Framework.

VSSD considers the software as an organisation of services to integrate the values into the SOA design. It considers three levels of abstraction. First, the *Abstract Level* describes the organisation at a high level of abstraction, where most of the contextual, domain-specific, information is left out. This level includes, for example, generic properties of organisations and an ontology to provide the meaning of the most primary concepts. Next, the *Concrete Level* takes into account the context of the organisation,

and describes the norms and rules. Lastly, the *Implementation Level* corresponds to the implementation process and contains the most detailed information.

3 Case Study

As illustration, we present an example of designing a web page for the Delft University of Technology to offer massive open online courses (MOOCs). From the requirements elicitation we got the (functional) requirement that we request users the permission to store information in a cookie. The cookie is required to provide a more user-friendly browsing experience; for example, the site could remember which courses you already followed. Let us assume that there is another requirement to provide support for impaired people; we could, for instance, include a text-to-speech functionality to read out loud parts of the website to people with visual impairments.

There are two hidden values at stake here: privacy and equality. The former, because the data we store in the cookies is of a personal nature, and consent has to be given before storing such data. The latter, because the inclusion of a text-to-speech functionality to assist impaired users is based on the idea that everyone should be able to use the website. Storing personal data while ensuring privacy also requires the upholding of another value: security. The relations between the software, the requirements and the underlying values is represented in figure 2 below.

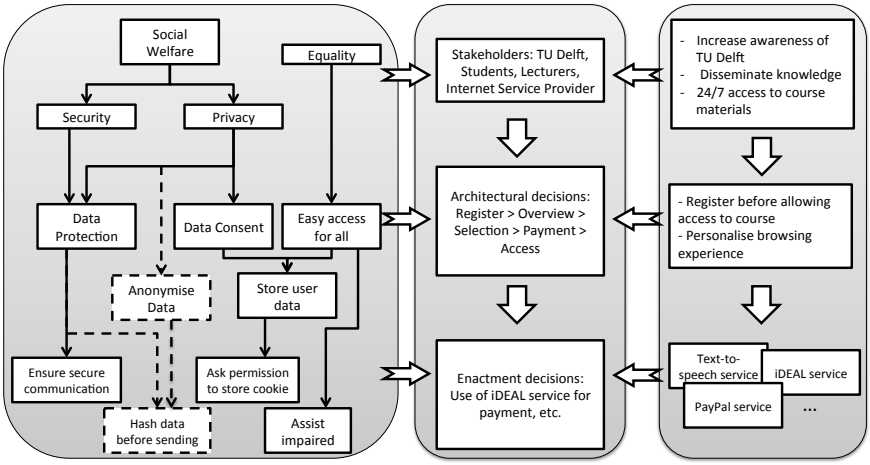

Fig. 2. VSSD applied to MOOC-page design.

The values, on the left-hand side of figure 2 represent the motivations and underlying values of the application. The requirements, and domain restrictions, are represented on the right-hand side of figure 2; they represent the manner in which the domain restricts the application design, which existing infrastructures (services) are available, and

in what way the objectives should be achieved. The middle part of figure 2 represents the implementation choices that end up in the runtime of the application. These are the characteristics of the software, describing concrete execution features of the application.

The strength of VSSD is in the explicit representation of the links between these layers (in figure 2, only shown explicitly in the vertical dimension of the Values View). Keeping explicit the motivations for particular design choices, either in the execution, or in the requirements, means alternative implementations can easily be explored without the need to redesign the application ground-up. The links explain the reasons why particular execution elements exist (because they implement a particular value) and/or why particular software behaviour was formulated (because that particular procedure was required by a high level norm or value). When changing the application, e.g., from a web page to an tablet app, these explicit links to the motivations can help you find the appropriate redesign steps. For example, in the redesign, instead of asking for cookies (which are not used on tablet apps), you could enable a secure connection for registration, and then perform all communication between the client (the tablet) and the server in an anonymised manner. Since the data stays on the tablet, and one can assume that the user owns that device (and is thus responsible for its security), these choices also implement the required value of privacy.

Since the above example is rather simple, the relation between the requirements and (hidden) values is quite straightforward. For more complex applications, with many more requirements, these relations might not be so obvious, and explicit links between the values, requirements and execution are direly needed to ensure the value sensitivity of the application.

4 Conclusions

The conceptual framework proposed in this abstract presents the first step into the explicit use of values in the design and implementation of systems, thereby achieving social responsibility in software. Next steps include the investigation and formalisation of the relations between the Views of the VSSD framework to create design procedures to guide the implementation and verification of value sensitive software. In future work, we will formalise the links present in figure 1, explore the verification of systems build with VSSD, and evaluate VSSD by modelling more use-cases.

References

1. Huib Aldewereld, Virginia Dignum, and Yao hua Tan. Design for values in software development. In Jeroen van den Hoven et al., editor, *Handbook of Ethics, Values, and Technological Design*. Springer-Verlag, 2015. forthcoming.
2. Batya Friedman, Peter H. Kahn, and Alan Borning. Value sensitive design and information systems. *Advances in Management Information Systems*, 6:348 – 372, 2006.
3. Ian Sommerville and Pete Sawyer. *Requirements engineering: a good practice guide*. John Wiley & Sons, Inc., 1997.
4. Ibo van de Poel. Tranlating values into design requirements. *Philosophy and Engineering: Reflections on Practice, Principles and Process*, pages 253–266, 2013.

5. Jeroen van den Hoven. Ict and value sensitive design. In Philippe Goujon, Sylvian Lavelle, Penny Duquenoy, Kai Kimppa, and Vronique Laurent, editors, *The Information Society: Innovation, Legitimacy, Ethics and Democracy In honor of Professor Jacques Berleur s.j.*, volume 233 of *IFIP International Federation for Information Processing*, pages 67–72. Springer Boston, 2007.
6. Olaf Zimmermann, Pal Krogdahl, and Clive Gee. Elements of service-oriented analysis and design: An interdisciplinary modeling approach for soa projects. *IBM developerWorks*, 2004.