Principles for Value-Sensitive Agent-Oriented Software Engineering

Christian Detweiler, Koen Hindriks, and Catholijn Jonker

Man-Machine Interaction Group, Delft University of Technology
{c.a.detweiler,k.v.hindriks,c.m.jonker}@tudelft.nl
http://mmi.tudelft.nl

Abstract. As software plays an increasingly important role in people's lives, the impact it has on their values frequently becomes apparent. Many software design methods address "soft issues", but very few address values explicitly. We present six principles that design methods should meet in order to properly deal with values. One area in which adherence to stakeholder values is important, is Agent-Oriented Software Engineering (AOSE). The Tropos AOSE method, with its concept of soft-goal, comes close to meeting our principles, but does not address values explicitly. Value-Sensitive Design is a methodology that does explicitly address value issues, but it offers little guidance in operationalizing them. We discuss a case study in which we attempt to capture values in Tropos' soft-goals after eliciting them using Value-Sensitive Design. Subsequently, we discuss to what extent Tropos adheres to our principles. Finally, we propose the introduction of values as a first-class entity in Tropos in order to meet our aims of dealing with values.

Keywords: Values, Value-Sensitive Design, Requirements Engineering, Non-Functional Requirements, Tropos.

1 Introduction

In 2009, the designers of the social networking website Facebook introduced a number of changes to the website. Due to these changes, users were no longer able to choose with whom they shared the list of people (their "friends") they were connected to on the website. Anyone logged in to Facebook could now see to whom any member of the website was connected. Further, Facebook decided users' profile pictures and the pages they "like"[1] were now publicly accessible information. That is, information users shared on the website regarding their interests would now be available to the Internet at large.

Facebook (partially) violated two values by introducing these changes. The changes violated users' value of autonomy by giving them diminished control over how their information is shared. Furthermore, the fact that certain personal information was now public impacted users' privacy. The violation of these values

[1] Web pages on Facebook about topics, people, places, books, etc. that people can "like".

led to user outrage and criticism by organizations such as Electronic Frontier Foundation [1]. Facebook responded that it tried to uphold the value of openness shared by their target audience [2].

This conflict of values and the way it became clear exemplifies the problem we seek to address in this paper. Designers necessarily impart social and moral values in making choices in the design of information systems [3]. That is, designers' values, such as openness, are "put into" software artifacts, albeit implicitly. Once a system has been put into use, it affects its stakeholders by supporting or hindering their values to various degrees. This ultimately affects the acceptability of information systems. Often, these values and value issues only become explicit after the software has been put into use, at which point the damage has been done. Therefore, we plead for dealing with values explicitly by treating them as separate "first-class entities" throughout the design process in software design methods.

This problem holds for software engineering in general, but is especially relevant in agent-oriented software engineering (AOSE), where we design agent-based systems. These systems are autonomous, reactive, pro-active, and have social ability [4]. Moreover, they act on stakeholders' behalf, so it is important that they meet stakeholders' requirements. Values can be considered requirements in that they are stakeholder needs that systems should uphold. The issue of meeting requirements is part of one of the areas of AOSE research identified in [5]. Weyns et al. conclude that we have to extend our research into goal-oriented design, verification and validation in order for agent-oriented software engineering to be adopted in industry. In particular, we should be able to provide guarantees with respect to stakeholder requirements.

In as far as design methodologies explicitly take values into account in the design process, it is in the form of non-functional requirements [6] or similar constructs. However, designers run the risk of leaving the impact values have on design implicit by representing values as non-functional requirements. Methods such as Quality Attribute Workshops and its notion of scenarios [7] and Attribute-Driven Design [8] deal with non-functional requirements formally, and have been applied to AOSE by Weyns [9]. However, AOSE methods typically neglect non-functional requirements. Values should be included explicitly as entities in themselves in order to be properly considered and to have an identifiable and justifiable effect on the design.

AOSE methodologies such as Tropos do focus on stakeholders' requirements (in the form of goals) throughout the design process, but do not explicitly take values into account. To address values in Tropos, they have to be represented as (soft)goals. Many important characteristics of values are lost in representing values as goals. Value-Sensitive Design [10] (VSD) provides a comprehensive framework for eliciting values, but provides little to make these values concrete. In this paper we propose to address this issue by combining elements of VSD with Tropos. This should form an AOSE approach that meets our aim of making the influence of values on the design explicit during all design phases.

This paper is organized as follows. In section 2 we briefly discuss the concept of values and discuss six value-related principles design methods should adhere to. We then discuss some common ways of dealing with such issues in requirements engineering. Then, in section 3 we discuss a case study to discover to which extent values can be dealt with in Tropos. In section 4, we analyze to what extent Tropos adheres to our principles and discuss important differences between values and the soft-goals we use to include values in Tropos, and propose introducing a value entity in Tropos. We draw conclusions and suggest directions for future work in section 5.

2 Values in Existing Software Engineering Methods

This section discusses to what extent values are already taken into account in existing software engineering approaches with an emphasis on agent-oriented software engineering methods, and in particular Tropos. Before this discussion, we present an overview of the concept of values and discuss the role of values in relation to the stakeholders and designers in the design of multi-agent systems. Finally, we provide a short introduction of the Value Sensitive Design method and discuss why VSD is not an answer in itself.

2.1 Values

The introduction describes a real world case of stakeholders' values (i.e., privacy, autonomy and openness) being hindered or supported by technology. Other values implicated in system design include human welfare, ownership and property, freedom from bias, and trust [10]. The general notions of norms and values are known to us all; norms and values are instilled into all of us during our childhood by our parents and social surroundings and continue to be throughout our lives.

Values are abstract (e.g., [11,12]), motivational constructs that apply across contexts and time [11]. They convey what is good (e.g., [13,14]) and important to us (e.g., [11,10]). For example, privacy was something good and important for users in the Facebook case. As a result, they were outraged when their privacy was not respected. They would have reacted similarly if another website, person or institution had failed to respect their privacy, as values hold across situations. As Hodges and Baron argue, values are convictions that some things ought to be and others not [15]. To make the concept of a value more precise it can be differentiated from similar concepts, such as laws, rules, goals, norms, standards, and so on (e.g., [12,13,16,14,17]).

Values have a special status due to their importance to their holders (violation of values is seen as deplorable or morally wrong) and the expectations they generate regarding the behavior of the holder and of others. Values create preference for behavior or action that supports them, which gives them a normative character. As Miceli and Castelfranchi point out regarding the normative character of values, “if something is good, it should be pursued” [13, p. 181]. For example, “honesty” is a value which gives rise to a norm “be honest”. Moreover, if something is good, it should not only be pursued by the holder of the value; it should be pursued by others as well. However, others do not always hold the same values. This normative character of values is a ground for conflicts when people hold different values or different priorities among their values.

Our work is concerned with the design of multi-agent systems and systems that are expected to have a social impact. Considering that the systems we build can conflict with the values and norms of the stakeholders of these systems, it is especially important to explicitly recognize the role values play in design.

Returning to the Facebook example, we can say that the value openness of Facebook gave rise to a norm of the Facebook team, i.e., "everybody should share personal information", which conflicts with the value of privacy of the users. In retrospect, could we not say that the way out of this conflict lies in considering the shared value of autonomy, with an associated norm that everybody should be able to decide for herself? Based on this shared norm we can derive the more specific norm that everybody has to decide for herself whether to share information or not. It is a compromise between openness and privacy that is acceptable to both developers and users of Facebook.

This example illustrates the abstract and normative nature of values. Values can be instantiated according to the situation at hand. For example, the value of autonomy is instantiated to insisting on control over how to share personal information on Facebook. The dormant problem of two conflicting abstract values (openness and privacy) became acute at the instantiated level. This leveled approach, working with instantiations, can also be found in the work of Maio [12]. To discover possible conflicts at an early stage of system development, we advocate value elicitation at the start of the project to make people consciously aware of their values; this will reduce costs, effort, and frustration. Proynova et al. make a similar plea [18], focusing mainly on elicitation.

We recognize that, though conflicts between moral values are not dealt with as such in the approaches described here, many mainstream software engineering methods do deal with conflicts of a similar form. Certain design decisions may hinder one value while achieving another. Conflicts with this structure are dealt with in mainstream software engineering methods in the form of tradeoffs between quality requirements (see, for example, [9]).

In our opinion the process of value elicitation at the start of a design process should answer the following questions. Which people's values can be impacted by the system under design and which people's values can impact the design of the system? In our view, this question is essential for the design of system and its answer is both obvious as well as treacherous by its obviousness. The answer to the first part of the question is the stakeholders, and the answer to the second part is the stakeholders and the designers/developers. The last addition, that of the designers, is easy to overlook, as the designers might unconsciously assume that their values are shared by the stakeholders. The Facebook example is illustrative of this point. We conclude that to avoid the negative consequences of violating values and to promote the support of values as much as possible, the following principles should be satisfied by design methods.

1. The values of all stakeholders including designers/developers should be elicited in as far as relevant for the system under design.
2. Stakeholder values should be addressed during all phases of the design process.
3. Conflicts between values of the designers and those of the stakeholders need to be discussed with those who issued the order for the system.
4. To account for the relevant values, to the relevant values need to be instantiated explicitly throughout the design process.
5. Design decisions can and need to be justified and evaluated in terms of explicit (instantiations of) stakeholders' values.
6. Conflicts between values need to be made clear and addressed in cooperation with the stakeholders.

These principles are used in the next section to discuss how existing requirement engineering methods as part of design methods deal with values.

2.2 Requirements and Values

Requirements engineering is one of the first steps in the larger process of software development. It is the process of identifying stakeholders and their needs, and documenting these in a form that can be analyzed, communicated, and subsequently implemented [19]. Broadly speaking, there are two types of requirements: functional requirements and non-functional requirements [6]. The former are requirements that define a function of the system, or something that a system will do. The latter define not what a system will do, but how it will do it. Requirements engineering has attention for "soft issues" such as politics and people's values, although dealing with soft issues is problematic as there is little guidance on how to do so [20]. Concepts used to specify soft issues include *non-functional requirements*, *quality attributes*, *soft constraints*, and *soft-goals*.

Though there is no consensus in the requirements engineering community as to exactly what non-functional requirements are [21], broadly speaking a non-functional requirement is "a software requirement that describes not what the software will do, but how the software will do it" [6, p. 6]. Non-functional requirements are often refered to as "-ities" or "-ilities" [22]. Examples of non-functional requirements include usability, maintainability, adaptability, efficiency, and flexibility.

The concept of non-functional requirement appears to be broad enough to cover values. In fact, some values, namely security and privacy (as a feature of security), have been dealt with in an extension of the Tropos method [23]. However, not all non-functional requirements are values. Non-functional requirements such as maintainability or portability, while important, are conceptually far removed from the moral good worth pursuing that values such as autonomy, trust, and justice point to. The examples of non-functional requirements given here are closely related to the envisioned system, whereas the examples of values are more closely related to humans, culture, or society. Furthermore, as far as we know, no specific guidelines exist for dealing with moral values in design methods that use the concept of non-functional requirements.

The related concept of *quality attribute* can be defined as "[a] feature or characteristic that affects an item's quality" where quality is understood as "[t]he degree to which a system, component, or process meets specified requirements" or "[t]he degree to which a system, component, or process meets customer or user needs or expectations" [24, p. 60]. As with non-functional requirements, this term is so general that it provides no guidelines for dealing with values specifically.

Soft constraints are requirements for dealing with over-constrained problems, as well as for dealing with uncertainty, vagueness or imprecision [25]. As stated in [25]: "They can be seen as a preferential constraint whose satisfaction is not required, but preferred." Treating soft constraints as "preferred but not required" disqualifies soft constraints as the way to model values as the moral wrongness of violating a value is lost. Nonetheless, we can try dealing with values as soft constraints. Soft constraints are to be elicited during the requirements engineering process, however, if values are not specifically addressed chances are that no values will be made explicit (principle 1). Soft constraints of stakeholders are typically taken into account, and that way principle 2 can be said to hold in as far as principle 1 is upheld. Principle 3 is not treated using values. Principles 4, and 5 are treated accepting that values are part of the whole set of soft constraints. Principle 6 is not dealt with as such.

Soft-goals, as used in e.g., Tropos [26], are requirements that are not clearly defined and do not have clear criteria for satisfaction, drawing on the notion of satisficing instead [27]. They are a form of non-functional requirements that refer explicitly to goals, an important concept in agent technology.

As we are particularly interested in agent-oriented software engineering [28] we focus on Tropos and its soft-goals. Treating values as soft-goals, we can summarize that principles 1, 2, 4, and 5 are treated to some extent in Tropos, but principles 3 and 6 are in no way part of the Tropos method. With respect to principle 1, indirect stakeholders are not taken into account, although the method could be easily adapted to cover this. Principle 2 is covered in the sense that soft-goals can play a role during all phases of the design. Principle 5 is covered in the sense that decisions are related to soft-goals, but not in as far as one soft-goal is weighed more heavily than another to make a choice.

Section 3 describes our effort to see how far we can get with modeling values as soft-goals in Tropos and will explain our conclusions regarding the principles.

Before focusing on Tropos and the possibilities soft-goals offer to include values in the design, we would like to mention one more approach that might be useful with respect to values.

2.3 Value-Sensitive Design

VSD "is a theoretically grounded approach to the design of technology that account for human values in a principled and comprehensive manner throughout the design process" [29]. In VSD, emphasis is given to supporting moral values or values with ethical import, such as human welfare, ownership of property, privacy, and freedom from bias [10].

VSD provides an iterative and integrative three-part methodology consisting of conceptual, empirical, and technical investigations. Conceptual investigations focus on discovering affected stakeholders, their values, and analyzing these values and tensions between them [30]. The first step is to perform a stakeholder analysis to identify direct and indirect stakeholders, which are the people who interact directly with the technology, and those who are impacted by the technology without interacting with it, respectively.

For each group of stakeholders, potential harms and benefits are identified. The list of harms and benefits can be used to map harms and benefits onto associated values, especially human values with ethical import.

Once these key values have been identified, a conceptual investigation of the values is conducted supported by (philosophical) literature, resulting in clear definitions of those values. Potential value conflicts, which can constrain the design space, are examined. Stakeholders are involved if conflicting values hinder one another in the design, such as accountability versus privacy.

Conceptual investigations need to be informed by empirical investigations of the technology's context. VSD does not prescribe a specific method for this stage, stating that "the entire range of quantitative and qualitative methods used in social science research is potentially applicable" [10]. Friedman and colleagues do suggest that semi-structured interviews of stakeholders can be a useful method to understand stakeholders' judgments about a context of use, an existing technology, or a proposed design.

Technical investigations focus on the properties and mechanisms of existing technologies that support or hinder human values. Alternatively, technical investigation can consist of designing a system to support identified human values. Though technical investigations of the first form and empirical investigations seem similar, technical investigations focus on the technology itself, and not on the individuals affected by it, as empirical investigation does. During this stage, it can be helpful to make explicit how design trade-offs map onto value conflicts and affect different groups of stakeholders.

It could be argued that, individually, the steps taken in VSD are common sense. Common sense as they may be, these steps are rarely taken together in a structured manner. As a result values are often neglected in design and addressed after the fact, as cases of privacy issues with social networking websites, bias in search engines, and intellectual property issues with file-sharing software illustrate. VSD offers a structured approach to addressing values.

The strengths of VSD lie in its focus on direct and indirect stakeholders, how they are or will be affected by the technology, and what values are implicated. The focus on a broad range of stakeholders, along with the identification of potential value conflicts and the aim to deal with values throughout design, suggest that VSD adheres to our six principles. However, VSD would benefit from means to not just elicit values, but actually incorporate them in design and eventually implement them.

3 Case Study: Values in Tropos

To discover to which extent values can be dealt with in Tropos in adherence to the six principles of Section 2.1 we performed a case study. The chosen case study is that of designing a conference management system with an emphasis on the values involved. We picked this case study as it was used in [28] to illustrate the use of three agent-oriented software engineering methods, including Tropos, and was based on an earlier case study presented in [31]. Furthermore, conference management systems are at the core of the peer-reviewing established by researchers to protect the quality of research. The decisions made during peer-reviewing have a high impact on researchers. Therefore, the design of such a system must be done in such a way that the norms and values of the stakeholders are respected as much as possible.[2]

The rest of this section is organized as follows. We first identify Tropos, we then describe the general purpose of conference management systems and identify the stakeholders, after which we inject the process of value elicitation for use later on. We then proceed with the remaining value-related steps in the Tropos method with an emphasis on how values are addressed in these steps.

The Tropos software development methodology supports the agent-oriented paradigm and the associated concepts of actors, plans and goals throughout the software development process [26], [28], [32]. Its main value-related steps are stakeholder identification, goal identification, and goal decomposition.

The general purpose of a conference management system depends on the stakeholders involved and vice versa. Tropos identifies stakeholders early in the design process, in the Early Requirements phase. The main stakeholders involved are a paper authors, paper reviewers, program chairs, and publisher of the proceedings [31]. To this we add the general public / government and the researcher as indirect stakeholders. We assume that the fundamental choice for blind peer reviewing has already been made in the organization of the conference. The general purpose is to support paper submission, bidding for papers for review, distribution of papers to reviewers, collection of reports, supporting program committee meetings, communication of results, and submission of camera ready versions of papers. All these aspects are subservient to the underlying concern of publishing high quality research only and blocking substandard research reports. The general purpose and the underlying concern already implicitly refer to a number of values.

Value elicitation was performed with each stakeholder group and ourselves as system designers. We used semi-structured interviews as suggested in the

[2] Note that the design of a conference management system in terms of the roles involved is primarily determined by the organization structure of the conference. In this case we chose for a conference management system that adheres to that of smaller conferences or workshops and ignored the more recent use of a Senior Programme Committee as is used in the AAMAS conference. It would be good practice to design the organization structure of the conference before designing the conference management system. However, for our purpose of showing how to deal with values, it is enough to start with some conference organization structure.

VSD method of [10]. In the interviews we explained the intention of designing a conference management system and described the basic activities it would support. We asked stakeholders to identify potential harms and benefits of such a system, and together with them identified the values underlying these harms and benefits. It is important to note that most interviewees had experience with existing systems and due to that it is likely they were reflecting on the systems they were familiar with. Also, most interviewees had experience with multiple stakeholder roles, making it difficult to rule out that they projected values they hold in one role to another role.

The authors mention anonymity of reviewers and conflicts of interests as potential harms and anonymity of authors as a benefit. They stated that anonymity removes context, which makes it difficult to assess reviewers' expertise and damages the quality of the discussion. Also, it allows reviewers to "ride their hobby horse", posing a threat to their objectivity. On the other hand anonymity of authors removes hierarchical considerations, leading to judgments based on quality and not on academic position. This is a potential benefit. The authors warned for conflict of interests arising from users occupying multiple roles within the same system. This could lead to reviewers who are also authors seeing the ranking of their own paper or reviewers reviewing papers of friends. The authors concluded that the harms are based on their values of transparency, fairness, and accountability, while the benefits are based on their values of fairness and privacy and would improve the quality of publications.

Reviewers mentioned anonymity of reviewers as a benefit. It also allows reviewers to be as critical as (they feel) they need to be. Together, we concluded that the underlying values are privacy and quality of publications. PC chairs considered reuse of the system across conferences to be a potential benefit, which contributes to the trustworthiness of the system. Trust is the underlying value here. A potential harm that one PC Chair identified was the potential for bias in seeing authors' names. This could lead to bias based on gender and ethnicity.

Publishers benefit from the peer review process the system supports. By publishing high-quality research and barring substandard research, the reputation of the publication and that of the publisher potentially increase, as do sales. This supports publishers' values of quality, profit and trust.

Researchers in general consider it a potential harm that poor quality research is disseminated. Poor quality research is damaging to the reputation of the research community with the general public and with government. Also, if researchers' own work is disseminated and of poor quality, it is damaging to their reputation with peers. Both senses of reputation, and the related value of scientific integrity, are values held by researchers.

The general public and government see the publication of high quality research and the barring of sub-standard level publications as potential benefits. These ultimately support the value of knowledge.

As system designers in this case, we discovered that we were influenced by our identification with the roles of author, reviewer, and PC chair, and as such shared many of the values of those stakeholders groups.

All stakeholders identified use of the conference management system for multiple conferences as a benefit. Reuse enhances the trustworthiness of the system and the process it supports. Also, the record of interactions with the system supports transparency and accountability.

In summary, we can see a range of values at stake here, among which potential or real conflicts exist, for example between transparency and privacy. This example conflict leads to opposing views on whether the system should provide anonymity. A compromise between such values must be found, that is, a feature that supports both or at least hinders neither.

Stakeholders' goals are identified next, and for every goal the developer decides whether the actor itself can achieve it or whether it needs to be delegated to another actor. Goals represent strategic interests of actors. A distinction can be made between (hard) goals and soft-goals. Hard goals have clear criteria for satisfaction. Soft-goals do not have clear criteria for satisfaction, drawing on the notion of satisficing instead [27].

The only option that Tropos has for representing the values identified in the previous stage are soft-goals. Due to space limitations, we will only discuss how the potential harm/benefit of anonymity, the potential harm of conflicts of interest and the underlying values at stake could be addressed in Tropos. Tropos actors are written in italicized bold. Goals and soft-goals are written in bold.

Authors saw the anonymity of reviewers as a potential harm as it prevents them from assessing the expertise of the reviewer. So, we could say the ***Author*** has a goal, **know reviewer identity**, which contributes positively to the values of **transparency** and **accountability**. The **know author identity** goal is why-linked to a goal dependency between the ***Author*** and the ***PC Chair***, **disclose reviewer identity**. We will discuss how this conflicts with ***Reviewers***' goals shortly.

Authors saw their own anonymity as a potential benefit. So, we introduce the goal **anonymity from reviewers**. This goal contributes positively to the ***Author***'s values of **privacy** and **fairness**, which we represent as soft-goals. The goal is why-linked to the goal dependency **protect author anonymity** between the ***Author*** and the ***PC Chair***.

Authors also saw conflicts of interest as a potential harm. So, the ***Author*** actor depends on the ***PC Chair*** to **avoid conflicts of interest**. This goal contributes positively to the value of **fairness**, represented as a soft-goal. However, since **avoid conflicts of interest** is a goal dependency and hence becomes the ***PC Chair***'s goal, the only option we have to link it to the ***Author***'s value of **fairness** in Tropos is the why-link.

Reviewers saw anonymity as a potential benefit. Therefore, we say that the ***Reviewer*** actor has a goal dependency, **protect reviewer anonymity**, on the ***PC Chair*** actor. This contributes positively to the ***Reviewer***'s values of **scientific integrity** and **privacy**, represented as soft-goals. Since the **protect reviewer anonymity** is a goal dependency, the only option we have to indicate the link between it and the values it contributes to is the why-link. However, the why-link is also a type of dependency, and only one link can be constructed for a

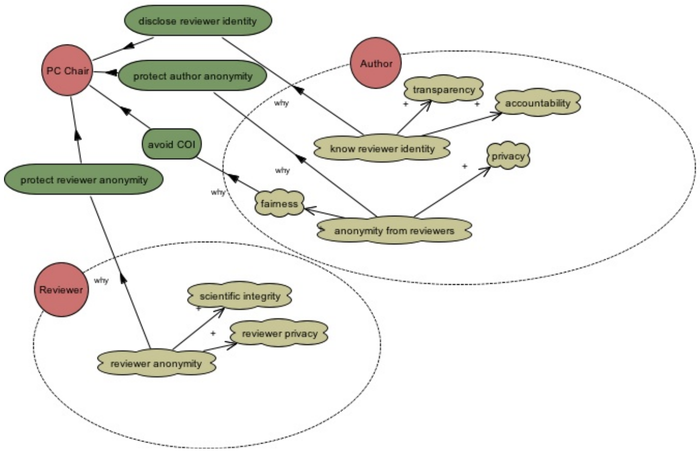

Fig. 1. Early requirements model of conference management system in Tropos with values as represented as softgoals

dependency. So, we have to define an intermediate goal, **reviewer anonymity**, which contributes positively to **scientific integrity** and **privacy** and is why-linked to **protect reviewer anonymity**.

Reviewers' goal **protect reviewer anonymity** obviously conflicts with authors' goal to **know reviewer identity**. Reviewers' value of **privacy** conflicts with authors' value of **transparency** here.

We attempted to model values in Tropos as soft-goals in order to meet the aims expressed in our six principles. However, there are a number of issues with this that we will discuss in the next section.

4 Discussion

4.1 Six Principles

We will now discuss the results of the case study described in section 3 in light of the six principles described in section 2.

The first principle states that the values of all stakeholders and designers or developers should be elicited as far as relevant to the system under design. While stakeholders are considered in Tropos, the group of stakeholders considered is limited to actors that will eventually use the system in some way. Indirect stakeholders, such as the general public in the case study above, are not considered, though they may be affected by the (output of the) system. Also, designers and developers are not considered in Tropos.

The second principle states that stakeholder values should be addressed throughout the design process. As the case study demonstrates, if we represent values as soft-goals in Tropos, then they can be said to be addressed throughout the design process. However, as we discuss below, values are not (soft) goals.

The third principle states that conflicts between the values of the designers and those of the stakeholders need to be discussed with those who issued the order for the system. Since Tropos does not consider the designers as such, conflicts between their values and those of the stakeholders do not become apparent.

The fourth principle states that values have to be instantiated explicitly throughout the design process. If we represent values as soft-goals, we can say that values are instantiated throughout the design process through the process of goal decomposition. However, there are problems with treating values as goals, which we discuss below.

The fifth principle is that design decisions need to be justified and evaluated in terms of explicit instantiations of stakeholders' values. We can say that goals and decompositions of goals into lower-level goals are design decisions. By drawing contribution links between these goals and soft-goals representing values, we can in a sense evaluate and justify these design decisions by seeing which design option (alternative subgoal) contributes best to the soft-goal (value) in question. It should be noted that the extent to which contribution can be expressed is limited. The metrics $+$ and $++$ indicate partial and sufficient positive contribution, respectively, and the metrics $-$ and $--$ indicate partial and sufficient negative contribution, respectively [26].

The sixth principle states that conflicts between values need to be made clear and addressed in cooperation with the stakeholders. In Tropos, the only links between (soft) goals are varieties of decomposition links, namely AND or OR decompositions, means-end links, or contribution links. Also, only one link can exist between these (soft) goals. That is, we cannot have a goal 1 contribute to a goal 2, and have that goal 2 contribute to goal 1. Therefore, we cannot express conflict between (equally abstract) values as such, for example openness and privacy. We could define a higher level soft-goal (value) and say that one lower-level soft-goal contributes positively to it, while another contributes negatively. These soft-goals would then be in conflict, in terms of how they contribute to the higher-level soft-goal, but this is not an option for intrinsic values (or ends) in conflict.

In summary, we can say that the first, fifth and sixth are satisfied to some extent; the second and fourth are satisfied if we consider values to be goals; the third cannot be said to be satisfied. However, this is the very reason why Tropos does not adhere to our principles. To adhere to the principles we would have to represent values as soft-goals, but values should not be treated as soft-goals.

4.2 Differences between Values and Goals

Values are not the same thing as goals. Miceli and Castelfranchi provide a useful distinction between these concepts. "Values are not goals, they are assumptions (more precisely, evaluations). A value is a judgment, though very general and vague. It says of something that it is good or bad. A goal is a regulatory state in someone's mind" [13, p.179]. They illustrate a further important feature of values in discussing the difference between values and norms: “Values in fact offer grounds for, or give rise to norms. Hence the normative’ facet of values: If something is good, it should be pursued” [13, p. 181]. If we represent values as soft-goals, the evaluative aspect (“X is good”) and the normative aspect (“X should be pursued”) are lost. Represented as a soft-goal, a value becomes something that can be satisficed (i.e., sufficiently satisfied). Not achieving a goal is not morally wrong as such. Violating a value is seen as morally wrong. This distinction is important. Not taking these aspects into account could lead to problems once the design has been implemented and put into practice, as we saw in the example of Facebook.

4.3 Dealing with Values in Tropos

Considering the issues with representing values as soft-goals, we propose some additions to the Tropos approach. First of all, in line with our first principle, we propose that the notion of stakeholder in Tropos be extended beyond those groups that delegate their goals to a system to all who will be affected by the system (i.e., direct and indirect stakeholders) and those who shape the system. These groups of stakeholders need to be approached as a source of requirements (values and otherwise) early in the requirements engineering process.

Second, since values should not be represented as goals, we propose the addition of a first-class value entity to Tropos. Since values are held by stakeholders, the value entity needs to be connected to the stakeholders that hold it. As we discussed above, values are general and abstract evaluations. They are conceptions of what is good and are important to their holder. We need to be able to indicate the goodness and importance of each value to its holder in some way, so we can prioritize values and assess the importance of addressing each one. Further, since what is good should be pursued, values can give rise to goals and norms. Hence, we need to be able to represent links between values and the norms and goals they generate. Norms should also be represented, but this is beyond the scope of this paper. Values eventually need to be implemented in some way. Antunes and Coelho’s Belief, Values, Goals (BVG) architecture uses values as central motivational mechanisms in their agents’ minds [33]. We see this as even more of a motivation to address values early on in design. Also, designers could make use of such an architecture to implement the values elicited and represented during the requirements phase.

Third, values and their instantiations can conflict. The conflict between Facebook’s value of openness and users’ value of privacy is a case in point. We need to be able to identify such conflicts in order to deal with them early on. To this end, we propose the addition of a conflict relationship between entities, specifically values, in Tropos.

Including indirect stakeholders as a source of (value) requirements, treating values as separate entities in models, explicitly representing conflicts between values, and dealing with values throughout design, as implementing our proposals will allow us to do, will provide us with an approach that adheres to the six principles described in section 2.1.

5 Conclusions

In summary, software impacts human values. In light of this fact and the special status values have, we proposed six principles designers should adhere to. Some requirements and software engineering concepts seem similar to values. However, there are some important differences between values and these concepts. VSD is a methodology that aims to account for (moral) values in design. VSD is a useful methodology for eliciting and defining stakeholders' values. However, VSD as-is does not provide a means for implementing such values. This makes it difficult to assess the extent to which values are incorporated in actual designs.

In our case study in we attempted to capture values in Tropos soft-goals and showed that Tropos as-is cannot fully handle our six principles. We argued that Tropos' soft-goals are fundamentally different from human values as described here. Representing values as soft-goals does not make values sufficiently explicit.

To address these problems, we propose complementing Tropos with a separate, first-class entity to capture values. This entity will allow the designer to explicitly represent values throughout the design process, and to make values concrete enough to operationalize them and to expose and address conflicts between them.

Future work should address the issue of representing values. Also, future work should deal with representing and addressing value conflicts, as these are an important source of many of the issues with values in design. To this end, a formal framework of values is needed. Further, the issue of dealing with different stakeholders' views on specific values should be addressed.

Acknowledgments. We would like to thank Danny Weyns, Marie-Pierre Gleizes, and the participants of the AOSE 2010 workshop for the suggestions they gave during a fruitful discussion there. Also, we would like to thank the anonymous reviewers for their helpful comments.

References

1. Bankston, K.: Facebook's new privacy changes: The good, the bad, and the ugly (2009)
2. Kirkpatrick, M.: Facebook's zuckerberg says the age of privacy is over (2010)
3. Friedman, B.: Human Values and the Design of Computer Technology. Cambridge University Press, CSLI, New York, Stanford, CA (1997)
4. Wooldridge, M., Ciancarini, P.: Agent-oriented software engineering: the state of the art. In: Agent Oriented Software Engineering III, pp. 55–82. Springer, Heidelberg (2001)
5. Weyns, D., Parunak, H., Shehory, O.: The future of software engineering and multi-agent systems. International Journal of Agent-Oriented Software Engineering 3(4) (2009)
6. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-functional requirements in software engineering (2000)
7. Barbacci, M., Ellison, R., Lattanze, A., Stafford, J., WeinStock, C., Wood, W.: Quality attribute workshops (qaw) (cmu/sei-2003-tr-016). Technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA (2003)

8. Wojcik, R.: Attribute-driven design (add), version 2.0 cmu/sei-2006-tr-023. Technical report, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, PA (2006)
9. Weyns, D.: Architecture-Based Design of Multi-Agent Systems. Springer, New York (2010)
10. Friedman, B., Kahn, P., Borning, A.: Value sensitive design and information systems. In: Human-Computer Interaction and Management Information Systems: Foundations, pp. 348–372. ME Sharpe, New York (2006)
11. Bardi, A., Schwartz, S.: Values and behavior: Strength and structure of relations. Personality and Social Psychology Bulletin 29(10), 1207 (2003)
12. Maio, G.R.: Mental representations of social values. Advances in Experimental Social Psychology 42, 1–43 (2010)
13. Miceli, M., Castelfranchi, C.: A cognitive approach to values. Journal for the Theory of Social Behaviour 19(2), 169–193 (1989)
14. Schroeder, M.: Value theory. In: Zalta, E.N., ed.: The Stanford Encyclopedia of Philosophy. Fall 2008 edn. (2008)
15. Hodges, B.H., Baron, R.M.: Values as constraints on affordances - perceiving and acting properly. Journal for the Theory of Social Behaviour 22(3), 263–294 (1992)
16. Rokeach, M.: Beliefs, attitudes and values: A theory of organization and change (1968)
17. Spates, J.: The sociology of values. Annual Review of Sociology 9(1), 27–49 (1983)
18. Proynova, R., Paech, B., Wicht, A., Wetter, T.: Use of personal values in requirements engineering–a research preview. Requirements Engineering: Foundation for Software Quality, 17–22 (2010)
19. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap, pp. 35–46. ACM, New York (2000)
20. Thew, S., Sutcliffe, A.: Investigating the role of'soft issues' in the re process. In: 16th IEEE International Requirements Engineering, RE 2008, pp. 63–66 (2008)
21. Glinz, M.: On non-functional requirements. In: 15th IEEE International Conference on Requirements Engineering, RE 2007, pp. 21–26. IEEE, Los Alamitos (2007)
22. Chung, L., do Prado Leite, J.: On non-functional requirements in software engineering. In: Conceptual Modeling: Foundations and Applications, pp. 363–379 (2009)
23. Mouratidis, H., Giorgini, P.: Secure tropos: A security-oriented extension of the tropos methodology. International Journal of Software Engineering and Knowledge Engineering 17(2), 285–309 (2007)
24. Ieee standard glossary of software engineering terminology. IEEE Std 610.12-1990 (1990)
25. Bartak, R.: Modelling soft constraints: a survey. Neural Network World 12(5), 421–432 (2002)
26. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems 8(3), 203–236 (2004)
27. Yu, E.S.K.: Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of the Third IEEE International Symposium on Requirements Engineering, RE 1997, pp. 226–235 (1997)
28. DeLoach, S., Padgham, L., Perini, A., Susi, A., Thangarajah, J.: Using three aose toolkits to develop a sample design. International Journal of Agent-Oriented Software Engineering 3(4), 416–476 (2009)

29. Friedman, B., Kahn, P., Borning, A.: Value sensitive design: Theory and methods. University of Washington Technical Report (2002)
30. Miller, J., Friedman, B., Jancke, G.: Value tensions in design: the value sensitive design, development, and appropriation of a corporation's groupware system, pp. 281–290. ACM, New York (2007)
31. Ciancarini, P., Nierstrasz, O., Tolksdorf, R.: A case study in coordination. In: Conference Management on the Internet (1998)
32. Giunchiglia, F., Mylopoulos, J., Perini, A.: The tropos software development methodology: processes, models and diagrams. LNCS, pp. 162–173 (2003)
33. Antunes, L., Coelho, H.: Redesigning the agents' decision machinery. Affective Interactions, 121–137 (2000)