A Value-Sensitive Approach to Agent-Oriented Software Engineering

Christian Detweiler, Koen Hindriks, Catholijn Jonker

{c.a.detweiler,k.v.hindriks,c.m.jonker}@tudelft.nl

Abstract. Prominent agent-oriented software engineering methodologies such as Tropos support the engineer throughout most of the development process. Though in this method attention is paid to system stakeholders by explicitly modeling them, potential harms and benefits of the system to these stakeholders, and the underlying human values that are impacted, are not explicitly accounted for. Value-Sensitive Design is a methodology that does address such issues, but offers little guidance in operationalizing them. We demonstrate differences between these methodologies using the development of a Conference Management System as a case study. Subsequently, we propose a means of integrating the methodologies to operationalize Value-Sensitive Design and take values into account in agent-oriented software engineering.

1 Introduction

In designing systems, designers make several choices as to which features to include and which to exclude, which users to support, and so on. They make choices on which functional and perhaps non-functional requirements to address. In the process they make judgments as to which of these features, users, or requirements would be best to address, which are less important, and which can be ignored. In making these choices, designers "necessarily impart social and moral values" [5]. Beyond such decisions, once a system has been put into use, it affects its direct stakeholders, the users, but also frequently impacts indirect stakeholders. Privacy issues with social networking websites [3], bias in search engines [11], and intellectual property issues with file-sharing software are a few examples of value issues brought about by technology.

Clearly, technological design, and its deployment and use, is a value-laden process, whether it be designer's values that steer their choices, or stakeholder's values affected by the eventual design artifact. However, there are few design methodologies that explicitly take values into account in the design process. Value-Sensitive Design [7] (henceforth VSD) provides a comprehensive framework for making value issues explicit in designing a system, but provides little in the way of actually implementing specific designs that take such value considerations into account and operationalize them. Several current agent-oriented software engineering (AOSE) methodologies do provide comprehensive means of designing agent-based systems, from early requirements specification through detailed design. However, additional support for explicitly taking human values into account is needed. There is a gap between value-based and agent-oriented approaches to design that we argue can be addressed by integrating elements of the VSD methodology into agent-oriented methodologies.

This paper is organized as follows. In section 2 we briefly discuss VSD and one of the more well-known and comprehensive AOSE methodologies, Tropos, to demonstrate that current state of the art AOSE methodologies could be improved by addressing value issues that arise from various norms and values that the stakeholders of a technical artifact have. Before we discuss a concrete proposal for integrating concepts of VSD into an AOSE methodology, we discuss a case study to identify issues related to values that are not explicitly supported by current agent-oriented software engineering methodologies in section 3. In section 4, we analyze important differences between central concepts in VSD and Tropos. In section 5, we present our proposal for an integration of VSD into AOSE, and conclude this paper in section 6.

2 State of the Art

There are currently several agent-oriented software methodologies that support and guide the process of designing a multi-agent system, such as Tropos, Prometheus, and GAIA (see [10] for an overview).

All of these methodologies support the MAS design process, but differ in support they offer for specific aspects of the design process or specific features. As our focus is on human values and the stakeholders that hold these values, we have selected one prominent methodology that – we believe – is representative of the state of the art, namely Tropos. Tropos seems very suitable for our purposes, as it includes specific support for stakeholder identification and modeling.

2.1 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"[6]. 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 [7]. Here moral refers to “issues that pertain to fairness, justice, human welfare, and virtue, encompassing within moral philosophical theory deontology, consequentialism, and virtue.” [7, p. 60].

Methodologically, it provides an iterative three-part methodology consisting of conceptual investigations, empirical investigations, and technical investigations.

Conceptual investigations focus on discovering values at stake and analyzing these and tensions between them[12]. Friedman and colleagues suggest beginning these investigations with either a value, technology, or context of use, depending on which is most central to the work at hand [7]. Inspecting the selected value, technology or context of use should reveal some value issues surrounding it.

The next 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 then identified. The list of harms and benefits can then 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 then examined. Stakeholders are involved if conflicting values hinder one another in the design, such as accountability versus privacy.

Several of the analyses of the conceptual investigation phase need to be informed by empirical investigations of the technology's context, and to evaluate particular designs. 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" [7]. Friedman and colleagues do suggest that semi-structured interviews of stakeholders can be a useful method to understand stakeholders' judgments about of a context of use, an existing technology, or a proposed design [7].

Technical investigations can either focus on the properties and mechanisms of existing technologies that support or hinder human values, or 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.

In summary, 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.

2.2 Tropos

The Tropos software development methodology supports the agent-oriented paradigm and the associated notions of actors, plans and goals throughout the software development process [4]. Tropos aims to cover the entire range of software development phases, from the Early Requirements phase, to the Late Requirements phase, the Architectural Design phase, the Detailed Design phase, and implementation [1, 9]. The emphasis placed on stakeholders in Tropos makes it closely related to, and therefore suitable for combination with, VSD.

In Tropos, the Analysis phase comprises both an Early Requirements and a Late Requirements phase. Tropos identifies stakeholders early in the design process, in the Early Requirements phase. After stakeholders are identified, stakeholders' goals are identified, and for every goal the developer decides whether the actor itself can achieve it or it needs to be delegated to another actor. Goals represent strategic interests of actors, and can be satisfiable hard goals, or soft-goals, which draw on the notion of satisficing [17]. Actors and goal dependencies between them are captured in an Actor Diagram.

The initial model of actors and dependencies is then expanded through goal decomposition. From each actor’s point of view, its goals are decomposed (through AND/OR, means-end, or contribution analysis) into sub-goals and plans. Non-functional requirements, represented as soft-goals, can also be identified at this point. The goal-decomposition process results in a Goal Diagram.

The Late Requirements phase expands on the Early Requirements phase by introducing the system-to-be as an actor. As in the Early Requirements phase, this process can be guided by analysis questions: *what are the goals that can be assigned to the system-to-be and which dependencies can be redirected from domain actors to the system?* [13]. The activities in this phase result in an extended domain model and the Late Requirements actor and goal diagrams, which represent stakeholders and their objectives, which can include non-functional requirements, as mentioned.

The Architectural Design phase builds on the context provided by the Early and Late Requirements models to define the global architecture of the system, consisting of actors (as sub-systems) and dependencies (as data and control flows).

The Detailed Design phase in Tropos involves analyzing and designing goal models of the sub-actors introduced in the Architectural Design phase. As it builds on the Architectural Design phase and refines the work products thereof, it is a suitable means of further specifying and eventually operationalizing values represented as soft-goals. The Detailed Design phase of Tropos has a number of work products, including agent goal diagrams, capability's activity and sequence diagrams, and computational representation of BDI concepts.

One of the strengths of Tropos, and of related agent-oriented software engineering methodologies (e.g. Prometheus [15]), is that it is highly structured and offers specific design artifacts or deliverables throughout the design process, such as actor and goal diagrams, architecture diagrams, etc. VSD does not dictate the creation of such specific design artifacts but it would be useful to do so to operationalize it in order to provide concrete guidance to designers.

3 Case study

To illustrate the use of the methodologies described here, we will discuss a case study that describes the real-world problem of designing a conference management system as introduced in [2]. As noted in [4] this case study illustrates a number of issues in (multi-agent) system analysis and design. Moreover, it involves a number of activities in which the values of the stakeholders of the system play a role, which makes this example useful and interesting from a VSD perspective as well.

The roles involved in the organization of a scientific conference described in [2] are a program committee (PC), which consists of chairs and members, paper authors, paper reviewers, and editor(s) of proceedings. The main activities involved are paper submission, bidding for papers for review, distribution of papers to referees, collection of reports, program committee meeting, communication of results, and submission of camera ready versions of papers.

As demonstrated in [4], the Tropos methodology can be applied to the design of a Conference Management System, through the Analysis, Architectural Design, and Detailed Design phases[1]. To our knowledge, VSD has not been applied to the case.

We will now describe the application of the VSD methodology to the Conference Management System case study, followed by a summary of the results of applying the Tropos methodology as discussed in [4]. To apply VSD to the Conference

[1] In that paper, the Tropos, Prometheus and O-MaSE methodologies were compared.

Management System case, we conducted semi-structured interviews with stakeholders, who had some experience with at least one of the roles mentioned in the original case. We explained the intention of designing a conference management system and described the basic functions it would support. In the interviews we then focused on eliciting the potential harms and benefits such a system posed to its stakeholders, and the related human values.

3.1 Value-Sensitive Design

In this case, the focus is on designing a new system and discovering the values of the stakeholders of that system. As such, using a particular value as a starting point, one of the options suggested in [7], does not apply here. What we do focus on is the design of a system to support the work involved in organizing a scientific conference. This system should support an existing work practice, which demands that we understand this work practice and the requirements it poses. Therefore, a technology and context of use are the most appropriate starting points here.

The process begins with stakeholder identification. Besides direct stakeholders such as reviewers, authors, and PC members already identified in the original CMS case, we identified *readers* as indirect stakeholders.

A wide range of potential harms and benefits to stakeholders was elicited in the interviews. Some related to conferences in general (and not to the system being proposed), others to human-computer interaction and interfaces, but many were value-related. In the examples we will discuss below, issues regarding the interface have been omitted.

Some of the most frequently mentioned potential harms were related to the anonymity of reviewers. It was stated that anonymity removes context, making it difficult to assess reviewers' expertise and damaging the quality of discussion. Also, it allows reviewers to "ride their hobby horse", posing a threat to their objectivity. The possible influence on one's own review of reading other reviews was also seen as a possible threat to objectivity.

Another frequently mentioned issue was that of scores for individual criteria not matching the overall score calculated by the system, due to opaque weighting of scores. This could lead to the rank of a paper not matching reviewers' evaluation of it, and to strategic scoring behavior. Interviewees also expressed concerns that PC members having access to the full text of submissions (to which they are not assigned as reviewers) introduces a risk of intellectual property theft.

Several interviewees also expressed concerns regarding the possibility of conflicts of interest introduced by users being able to occupy multiple roles within the same system. For example, PC members that are also authors could see the ranking of their submission, or reviewers could review papers of colleagues.

Potential benefits were also mentioned. The most frequently mentioned benefits were related to anonymity. Several interviewees stated that anonymity removes hierarchical considerations, leading to judgments based on quality and not on academic position. It also allows reviewers to be as critical as (they feel) they need to be.

Many saw the possibility of using the same CMS for multiple conferences as a potential benefit, as it enhances the trustworthiness of the system and the process it supports. Also, the record of interactions with the system that could be provided enhances the transparency and accountability.

A number of interviewees considered the ability of reviewers to review others' reviews to be beneficial to learning to review. Some went beyond that, stating that openness of the entire system would be beneficial to those using it.

In [7], it is suggested that identified harms and benefits be mapped to underlying values. Though anonymity can be a value in itself, here the potential harms associated with it are related to transparency, objectivity, and accountability. Issues with the scoring and ranking of papers can be linked with transparency and fairness. The potential harm of intellectual property theft can, of course, be mapped to the value of intellectual property. The issues surrounding multiple roles and conflicts of interest can be mapped to the value of fairness.

As for potential benefits, those to be gained from anonymity can be mapped to fairness. The potential benefits of a single CMS across multiple conferences can be mapped to trust. As mentioned, the record of interactions such a system would make possible can be linked to transparency and accountability.

Strengths and weaknesses

The application of the VSD methodology to the Conference Management System case illustrates some of the strengths and weaknesses of VSD. As mentioned, one of the key features of the VSD methodology is that it allows the designer to identify and make explicit stakeholders and their values, assess how the technology will affect them, and identify which values are implicated. For example, we found that there is a potential harm of conflicts of interest, which implicated the value of fairness.

A weakness of VSD, also illustrated by the case study here, is that the methodology as-is does not contain steps that lead to an actual design or system specifications of any kind. In a sense, VSD is a means for collecting (early) requirements. It provides some heuristics (through empirical and technical investigations) for evaluating a system designed based on these conceptual investigations, but remains unclear as to how, specifically, to embody identified values in a system or translate values into system features. VSD would benefit from extension by a method that handles such a translation of abstract, high-level concepts into concrete system features. We believe that combining VSD with Tropos may provide a suitable approach towards this end.

3.2 Tropos

As described above, Tropos deals with several phases of design, from the early requirements phase through implementation, and as such it translates abstract, high-level requirements into concrete system features. Here, we will focus on the Analysis phase, consisting of Early and Late Requirements phases, as the level of abstraction in this phase is most closely related to that of VSD as described above. In the Conference Management System case described in [4], application of the Tropos methodology begins in the Early Requirements phase with the identification of stakeholders, namely paper authors, conference program committee, program committee chair, paper reviewers and proceedings publisher. These are then represented as the actors **Author**, **PC**, **PC Chair**, **Reviewer**, and **Publisher**.

Subsequently, stakeholders' goals are identified and the designer determines whether it is a goal the actor can achieve on its own or a goal the actor needs to delegate to another actor. For example, the **Author** actor has the **Peer review** goal. Obviously, this is not something the **Author** can do himself or herself, so it is delegated to the **PC** actor, creating a goal dependency between **Author** and **PC**. These actors and goal dependencies are captured in the Early Requirements Actor Diagram (using the TAOM4e tool[2]).

Goal identification is followed by (AND/OR, means-end, or contribution) decomposition. For example, the **Manage Conference** goal dependency between the **PC Chair** and **PC** actors is AND-decomposed, from the **PC Chair** actor's point of view, into the sub-goals **Get papers**, **Select papers**, **Print proceedings**, **Nominate PC** and **Decide deadlines**. These are visually represented in an actor goal diagram, which includes soft-goals in which non-functional requirements can be captured. For example, the actor goal diagram for the **PC Chair** actor contains the soft-goal **Conference quality**, to which the soft-goal **better quality papers**, among others, contributes positively [4].

In the Late Requirements phase in Tropos, the system-to-be is introduced as an actor. In the case presented here, this actor is the Conference Management System (CMS) system (as opposed to human) actor to which goals are assigned and dependencies are redirected. For example, the PC delegates the **Coordinate conference** goal to the CMS System actor. This goal is then analyzed from the **CMS System** actor's point of view and AND-decomposed into sub-goals, such as **manage submissions**, **manage reviews**, and **manage decision**. Operative plans are then specified and linked to goals, forming a means-ends relationship, such as that between the **accept** and **reject** plans of the **manage decision** goal[4]. All of this is captured in the extended domain model and Late Requirements Diagrams. The Formal Tropos language [8] can be used at this stage to specify formal properties of entities in the model. Design is then continued in the Architectural Design and Detailed Design phases of Tropos, which will not be discussed here.

4 Analysis

Based on the two case studies described above, we can see some similarities between VSD and Tropos, especially in VSD's conceptual investigations and Tropos Early and Late Requirements phases or Analysis phase. The most obvious overlap consists of their mutual focus on stakeholders. Both start off with identifying stakeholders. Both are essentially stakeholder-centered in this initial phase; after identifying stakeholders, VSD proceeds to identify their values, and Tropos identifies their goals. A further similarity is that both VSD and Tropos examine an existing practice in which a system will be introduced, through focus on context of use and actor networks into which a system is introduced, respectively. At first glance, another overlap seems to exist between VSD's values, such as fairness, and Tropos' soft-goals, such as quality of papers. However, there are some important differences to consider.

[2] http://sra.itc.it/tools/taom4e

Though both methodologies begin their analyses by asking who the stakeholders are, they differ in their interest in these stakeholders. VSD proceeds to ask how stakeholders are or will be affected by the technology, and what values are implicated. Tropos, on the other hand, follows stakeholder identification by asking what stakeholders' goals are and what the dependencies between these stakeholders are for achieving them. Later in the analysis phase in Tropos, the system is introduced as an actor to which stakeholders' goals can be delegated. This difference in aims is important for distinguishing between values and goals.

As mentioned above, VSD emphasizes supporting moral values or values with ethical import. An assumption of VSD is that certain values are held universally (though they can play out in various ways in specific cultures) [7]. Many people can hold the same value, and expect others to respect that value. Respecting values has to do with promoting what supports them and avoiding what hinders them. Here promotion and avoidance are both values in themselves, but in an instrumental way in that they are a means of bringing about the value that they respect.

In a practical sense, a value identified through the application of VSD can be seen as a global design requirement for a good (in the ethical sense) system, which gives rise to (or is a source of) value-based qualities of or constraints on a system.

In Tropos, no such claims are made about goals. Goals refer to actors' strategic interests [1]. They are actor-specific. Goals can be hard goals, which are satisfied, or softgoals, which have no clear-cut criteria for achievement and are satisficed. If a goal is not satisfied or a soft goal is not satisficed, we cannot necessarily say that that is morally wrong. Goals can be achieved by the actor itself by means of further sub-goals and plans. If the actor cannot achieve a goal itself, it can delegate it to another actor, creating a goal (or plan, or resource) dependency between the two actors. The aim of capturing stakeholders' goals is ultimately to delegate (some of) those goals to a system and decompose them into specific system functions.

So, the concepts of values in VSD and goals in Tropos differ in scope (values are global, whereas goals are actor-specific); in contextual role (values are something to be respected, whereas goals are something to be achieved personally, or delegated); in their role in a design (values eventually lead to value-based qualities and constraints, whereas goals eventually lead to functions as operational plans); and in their ethical load (the type of values here concern moral good, goals do not necessarily). As such, values cannot be captured directly in Tropos goals or soft goals. This is not to say, however, that values should not be taken into account. On the contrary, many of the identified values concern what is considered morally good. Taking these values into account should lead to systems that perform well in a functional sense as well as in a moral sense. Identification of these values and their distinctness from goals is not enough, though. What is needed is a way of integrating the concept of values into the Tropos methodology. An approach to this is described in the following section.

5 Value-Sensitive Agent-Oriented Software Engineering

As we have discussed, the concept of goals in Tropos, both hard and soft, is different from that of values in VSD in a number of important ways. To complement VSD with means for incorporating values in design, and to complement Tropos with a means to account for stakeholders' values, we propose a number of extensions to the Tropos methodology. This brings VSD closer to its goal of embodying values in technology and allows Tropos to capture a wider range of important stakeholder interests.

The first of these is a set of "design questions" that help designers using Tropos to identify values and distinguish them from "normal" goals. A number of similar questions are already offered as a guide to starting off the Early Requirements phase in work on Tropos (e.g. [13]), such as "who are the stakeholders?"

These questions can be enhanced with questions suggested in work on VSD (e.g. [7]). How will stakeholders be affected by the technology? What values are implicated? The first question leads to the identification of potential harms and benefits. The second leads to mapping the identified potential harms and benefits to (underlying) values. For example, an author (and, arguably, other stakeholders) may be affected if the CMS distributes his or her paper to a reviewer he or she knows, such as a reviewer that is also a co-author. Such a conflict of interests constitutes a potential harm, which is undesirable. We can formulate this as a value, namely "avoid potential harm". This value is then instrumental (or a means) to the higher-level that is implicated. The implicated value, in this case, is fairness (or freedom from bias).

To assess whether something is a value or a goal, the designer can ask the following questions, which are derived from the analysis in section 4. *Is this requirement actor-specific or global? Can it be achieved by an actor? Can this requirement be fulfilled with sub-goals or plans, or does it lead to qualities of and constraints on the system? Can not meeting the requirement be considered wrong in a moral sense?*

In this case, fairness is a global requirement, not actor-specific; it is not something an actor alone can achieve or delegate, it is something to be upheld generally; it cannot readily be fulfilled with a specific sub-goal or plan, it calls for constraints on the system; not meeting it, creating an unfair system, can be considered morally wrong.

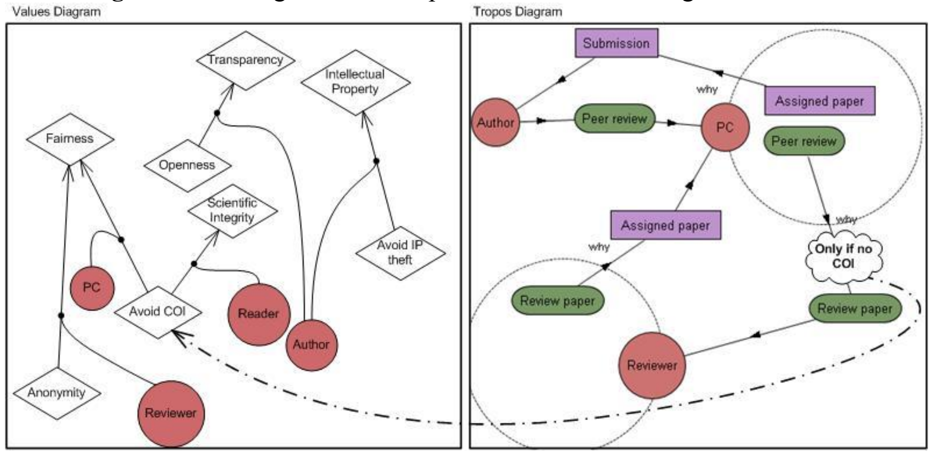
Now that values can be identified and distinguished from goals, they must be represented in Tropos models to be able to properly take them into account. As discussed above, values cannot be captured directly in Tropos goals or soft goals.

So, new entities will need to be introduced to Tropos models. We propose the addition of Value Diagrams to Tropos (and TAOM4e), with two new model entities: the value and actor-value link. The value entity can be either a high-level ("end") value or an instrumental ("means") value. The actor value link links an actor to a means-end link between a high-level and instrumental value, to indicate for which actor it is important that the instrumental value and related high-level value be respected. The actor-value link is linked to the means-end link between instrumental and high-level value, because high-level values can have multiple associated instrumental values, and instrumental values can be a means to multiple high-level values. The benefit of such a value diagram is that it provides an overview of values that come into play in the system design, which direct and indirect stakeholders they matter to, and how these values are linked to system value based system qualities and constraints.

In the figure depicted below, values are represented as parallelograms in a diagram that illustrates what a Values Diagram looks like. This diagram is separate from the Actor and Goal diagrams already present in Tropos. Within the scope of the design, these high-level values are seen as ends. Potential harms and benefits hinder or support the “end” value. As the basic aim is to uphold “end” values, avoiding harms and promoting benefits can be considered instrumental values. Therefore, instrumental values for a design can be derived from identified harms and benefits. These instrumental values generate value-based constraints (to implement avoidance-values) and value-based qualities (to implement promotion-values) in the Tropos model. We use a notion of qualities and constraints put forward in work on Secure Tropos [14].They state that “[q]ualities are properties or characteristics of the system that its stakeholders care about, while constraints are restrictions, rules or conditions imposed to the system and unlike qualities are (theoretically) non-negotiable.” [14, p. 1389]. As in that work, constraints are represented as clouds, and qualities as soft goals, with the addition of links to the values from which they are derived. This link indicates that such constraints and qualities are value-based, and as such must be accounted for.

In our example, the value **fairness** is hindered by the potential harm of a conflict of interests (COI). As mentioned, potential harms should be avoided and potential benefits promoted, so an instrumental value, **Avoid COI** is introduced. As it is an “avoid-value”, a constraint must be introduced in the Tropos model to handle this value. We introduce a constraint, **Only if no COI**, on the **Review paper** goal dependency between the PC actor and the Reviewer actor. Other values in the Values Diagram have not been incorporated into the Tropos Early Requirements Diagram in this example.

Fig. 1. Value Diagrams and Tropos Actor and Goal Diagrams for COI



6 Conclusions and future work

State of the art agent-oriented software engineering methodologies, such as Tropos, provide system designers with a large amount of tools to capture and analyze requirements and translate these into a multi-agent system design. However, they do not explicitly take human values into account. VSD does provide a way to consider human values affected by a technology, but lacks a comprehensive way to incorporate such considerations in a design.

In this paper, we have demonstrated how the application of the VSD methodology makes explicit direct and indirect stakeholders, potential harms and benefits to these stakeholders, and their underlying values. However, there is a lack of design artifacts or deliverables that VSD in itself produces.

The Tropos methodology, on the other hand, does provide such deliverables. Furthermore, it explicitly takes stakeholders into account, and is in this regard closely related to VSD, even though Tropos does not explicitly identify indirect stakeholders. At first glance, Tropos seems capable of capturing values as soft-goals, but we have argued that, upon considering the differences in aim, implication, and ethical load between values in VSD and goals in Tropos, goals (even soft-goals) are not an adequate means of capturing values. Also, potential harms are not explicitly taken into account from the onset in Tropos.

It appears, then, that the methodologies described here complement each other in useful ways. We argue that integrating value-sensitive and agent-oriented approaches is a natural step forward.

By using design questions suggested in previous work on VSD, designers can identify potential harms and benefits, and underlying values. We expanded these guiding questions, with questions to help designers make a distinction between values and goals. To incorporate these additional requirements into Tropos models, we propose the addition of two new entities to Tropos, the value and the actor-value link. These are to be represented in a Values Diagram, which shows high-level values, their instrumental values, and actors (including, importantly, indirect stakeholders) to which they are important. Instrumental values lead to value-based qualities and to constraints. The value-based qualities and constraints in Tropos Actor and Goal diagrams are linked to their source values, which indicates that they ought to be dealt with. With these complements to the Tropos methodology, important first steps will be made in the directions of value-sensitive agent-oriented software engineering.

Much remains to be done along these lines, though. What has been presented in this paper is a proof of concept. For the proposed extensions to be put into practice, the identification steps and model entities will need to be supported by a formal language and tools such as TAOM4e. Also, methods will be required for testing and evaluation of designs.

Another important issue, mentioned in our discussion of VSD, but not dealt with in our proposed extension of Tropos, is that of conflicts between values. Though work has been done on resolving conflicts between goal-based requirements (e.g. [16]), values, as we have argued, are distinct from goals. Conflict resolution methods for goal-based requirements may not be suited for resolving conflicts between value-based requirements. A first step in this direction in requirements analysis would be to explicitly identify such conflicts and represent the fact that value conflicts exist in requirements models.

References

1. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F. and Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems, 8(3):203-236, (2004)
2. Ciancarini, P., Nierstrasz, O. and Tolksdorf, R.: A case study in coordination: Conference management on the internet. ftp://ftp.cs.unibo.it/pub/cianca/coordina.ps.gz (1998)
3. Debatin, B., Lovejoy, J.P., Horn, A.K. and Hughes, B.N.: Facebook and Online Privacy: Attitudes, Behaviors, and Unintended Consequences. J Comput-Mediat Comm, 15(1):83-108, (2009)
4. DeLoach, S., Padgham, L., Perini, A., Susi, A. and Thangarajah, J.: Using three AOSE toolkits to develop a sample design. International Journal of Agent-Oriented Software Engineering, 3(4):416-476, (2009)
5. Friedman, B. (ed.): Human Values and the Design of Computer Technology. Cambridge University Press and CSLI, New York, NY and Stanford, CA (1997)
6. Friedman, B., Kahn, P. and Borning, A.: Value sensitive design: Theory and methods. University of Washington Technical Report, (2002)
7. Friedman, B., Kahn, P. and Borning, A.: Value sensitive design and information systems. Human-Computer Interaction and Management Information Systems: Foundations. ME Sharpe, New York:348-372, (2006)
8. Fuxman, A., Liu, L., Mylopoulos, J., Pistore, M., Roveri, M. and Traverso, P.: Specifying and analyzing early requirements in Tropos. Requirements Engineering, 9(2):132-150, (2004)
9. Giunchiglia, F., Mylopoulos, J. and Perini, A.: The tropos software development methodology: processes, models and diagrams. Lecture Notes in Computer Science:162-173, (2003)
10. Henderson-Sellers, B. and Giorgini, P.: Agent-oriented methodologies. IGI Global (2005)
11. Introna, L.D. and Nissenbaum, H.: Shaping the Web: Why the politics of search engines matters. Inform Soc, 16(3):169-185, (2000)
12. Miller, J., Friedman, B. and Jancke, G.: Value tensions in design: the value sensitive design, development, and appropriation of a corporation's groupware system. *In Proc.* pp. 281-290. ACM (2007)
13. Morandini, M., Nguyen, D., Perini, A., Siena, A. and Susi, A.: Tool-supported development with tropos: The conference management system case study. Lecture Notes in Computer Science, 4951:182, (2008)
14. Mouratidis, H., Giorgini, P. and Manson, G.: An ontology for modelling security: The tropos approach. Lecture Notes in Computer Science:1387-1394, (2003)
15. Padgham, L. and Winikoff, M.: Prometheus: A methodology for developing intelligent agents. Agent-Oriented Software Engineering Iii, 2585:174-185, (2002)
16. Van Lamsweerde, A.: Goal-oriented requirements engineering: A guided tour. *In Proc.*, Vol. 249. pp. 263. Toronto, Canada (2001)
17. Yu, E.S.K.: Towards modelling and reasoning support for early-phase requirements engineering. Re '97 - Proceedings of the Third Ieee International Symposium on Requirements Engineering:226-235, (1997)