2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology

A Normative Agent System to Prevent Cyberbullying

Tibor Bosse and Sven Stam
Vrije Universiteit Amsterdam, De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
tbosse@few.vu.nl, sven.stam@gmail.com

Abstract — **Automated approaches to prevent the negative effects of cyberbullying mainly focus on affective agents that provide support for victims. The current paper takes a complementary approach, which attempts to minimise the amount of occurrences of cyberbullying in the first place. The approach consists of a system of normative agents, which are physically present in a virtual society. The agents, which reason based on a BDI-model, use a number of techniques to detect various norm violations, including insulting and following. By using rewards and punishments, they try to reinforce the desired behaviour of the users. The system has been implemented and tested within a virtual environment for children between 6 and 12 years old, called Club Time Machine. In a real world experiment, the behaviour of the users of the virtual environment has been logged and analysed by means of a logic-based checking tool. The results show that the normative agents have the potential to reduce the amount of norm violations on the long term.**

Keywords - cyberbullying, normative agents, BDI.

I. INTRODUCTION

In parallel with the rapid development of electronic communication technologies in recent years, a new type of bullying has emerged. This type of bullying, often referred to as cyberbullying, is defined in [2] as “the general term describing any communication activity using cyber technology that could be considered harmful to individual or collective well-being”. Like normal bullying, cyberbullying may involve predation, hate group recruitment, invasion of personal privacy, harassment, stalking and harmful speech and behaviour. As a result of the increased use of electronic communication technologies, various different forms of communication have started to appear: instant messaging, text message, forums, social communities, virtual societies, and so on. Since these types of media are relatively easy to access, but difficult to monitor, cyberbullying has become a major societal problem [12].

To date, most types of cyberbullying are still very hard to prevent. For instance, research reported by [18] shows us that Internet filters have a very marginal effect on reducing unwanted access to websites like pornography sites. Bamford [2] claims that the only real solution to the problem is the intervention of the parents and teacher. However, since the majority of the Internet users is under the age of 25, this presents a problem, because parents and teachers are not as familiar with the Internet and its possibilities as their children. It is therefore hard for them to check if their children are involved in bullying activities.

This paper proposes an alternative approach to cyberbullying: we present a system composed of multiple agents that control users’ norm adherence within virtual societies. Being physically present in the virtual society, the agents continuously monitor the behaviour of the visitors, communicate with each other to maintain shared beliefs of the visitors’ characteristics, and apply punishments and rewards to influence their behaviour.

The remainder of this paper is structured as follows. Section 2 presents a brief overview of the state-of-the-art in approaches to prevent cyberbullying, and normative agent systems. Section 3 introduces the agent-based system proposed in this research at a high level. Next, Section 4 describes the engine to recognise the user’s behaviour in detail, and Section 5 describes the behaviour of the individual agents. Section 6 describes a preliminary experiment that has been performed to evaluate the system, as well as its results. Section 7 is a discussion.

II. RELATED WORK

Because the problem of cyberbullying is relatively young, only few approaches exist that try to deal with this. As mentioned above, traditional approaches focused at increasing supervision of parents and teachers, or at increasing children’s awareness of the concept and negative consequences of cyberbullying by means of working groups and other types of propaganda. Also various forms of peer support have been introduced, to help victims of cyberbullying to deal with their negative experiences [7]. Only recently, researchers have started to automate some of these approaches. For example, systems have been developed based on embodied conversational agents that provide peer support for cases of (cyber)bullying by showing affective (e.g., empathic) behaviour and giving advice (see, e.g., [1, 16]).

Indeed, these systems have shown to be successful in reducing the negative effects of cyberbullying on children’s mood and behaviour. However, these effects can be reduced even further if the extent to which cyberbullying takes place is limited in the first place. This is the main aim of the current paper, i.e., not to support cyberbullying victims, but to prevent cyberbullying (to some extent) from happening.

To this end, a multi-agent system is developed, consisting of a number of normative agents, which are physically present in a virtual society. The agents exploit several intelligent techniques to detect (different types of) norm violations. This approach is similar to the work presented in [11], which proposes the use of machine learning techniques to identify harmful entries on websites, such as slanderous or abusive messages. The current system goes one step further, by not only detecting norm violations, but also by preventing them via a mechanism based on punishments and rewards. These measurements are taken by virtual agents that are literally walking around in the virtual societies.

Regarding norm violations, a wide sub-community exists that focuses on the development of *normative multi-agent systems*, i.e., systems composed of multiple agents whose behaviour is controlled by a set of formal norms. Examples of such normative systems are NMAS/SMART [9], ISLANDER [14], OMNI [8], and MAS-SOC [10]. In contrast, the agents presented in the current paper do not have to adhere to norms themselves, but will enforce norm adherence in other (human) agents in a virtual society. Nevertheless, the presented approach has been partly inspired by the research in normative multi-agent systems.

III. System Overview

In this section, a global overview of the normative agent system is presented. First, Section 3.A briefly introduces the virtual environment in which the system will be applied. Next, Section 3.B describes the system architecture, and Section 3.C explains how the different agents in the system interact. More details about the system can be found in the M.Sc. Thesis in [13].

A. Virtual Society

In order to test the normative agent system in a realistic context, a collaborative virtual environment is required in which human users can interact by means of physically embodied avatars. A suited collaborative virtual environment for this purpose is Club Time Machine. Club Time Machine is a virtual environment created for children between the age of 6 and 12, where they can be entertained, educated and interact socially with other children (see also Figure 3 later on). This virtual environment is suited because of two reasons. First, the users in the virtual environment interact socially, so there are social norms the agent system can monitor. Second, its target group is children, which is a group in which bullying activities are quite common.

B. System Architecture

A global overview of the system architecture is shown in Figure 1. At the top level, two components are distinguished: the Club Time Machine (CTM) environment and the normative multi-agent system (NMAS). CTM is based on the SmartFox server, a massive multiplayer platform for developing games. Actions performed by the visitors of C (e.g., movements or text entries) are communicated by the SmartFox server, and then interpreted by NMAS. In response to this, NMAS generates appropriate actions (e.g., punishments or rewards) by the agents, which are sent back to CTM, where they are executed.

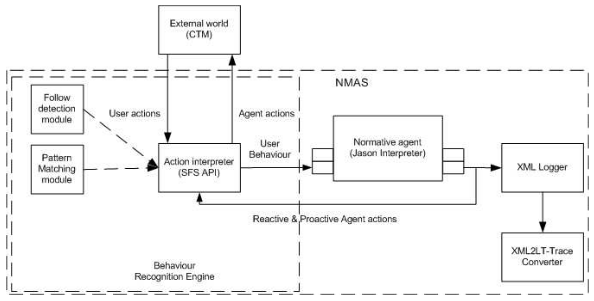

Figure 1. Framework Architecture.

One level lower, NMAS consists of four sub-components: the Behaviour Recognition Engine (BRE), the Normative Agents, an XML Logger, and an XML2LT-Trace Converter. BRE interprets the raw user actions as received from the SmartFox server, and translates them to intuitive high-level behaviours such as ‘insulting’ or ‘helping’ (see Section 4). As shown in Figure 1, this system currently makes use of two task-specific modules, but additional modules can easily be plugged in. The output of BRE (i.e., the high-level behaviour of the users) is transferred to the Normative Agents, which reason about it in order to determine which actions are appropriate (see Section 5). To this end, they make use of the BDI-based programming language AgentSpeak and its Java-based interpreter Jason [3], similar to the approach presented in [10]. Note that the normative agents are represented in Figure 1 as one component, but in reality this component comprises a whole team of interacting agents (see Section 3.C). The output of the Normative Agents (i.e., the concrete actions to be performed) is transferred back via BRE to CTM. In addition, all intermediate steps in the Normative Agents’ reasoning processes are stored (by the XML logger) in XML files. This is done to enable off-line analysis of the system’s behaviour. Each reasoning cycle, the XML logger writes the conclusions that are derived by executed reasoning rules into the XML file. Finally, the XML2LT-Trace Converter converts the XML log files into so-called LEADSTO (LT) traces. These traces, which have the form of time-stamped sequences of events, can be read by the LEADSTO/TTL verification tool [4, 5]. As will be explained in Section 6, this tool can be used to check dynamic properties of the log files in an automated manner.

C. Interaction between Multiple Agents

The virtual environment used by CTM consists of a large number of separate rooms. For this reason, monitoring the behaviour of all users in each room would be a very difficult task for a centralised system. Instead, it is more efficient to take a distributed approach. That is, to make every agent responsible for monitoring a small part of the entire environment. Thus, instead of having one single agent that patrols through the entire environment of CTM, each room is populated by one normative agent. All of these agents communicate their findings to each other and cooperate in order to enforce the desired behaviour of the users.

More specifically, based on the actions of a user, each normative agent has the ability to punish or reward that user and to update its own beliefs about the reputation of that user, via the mechanism explained in Section 5. Hence, when a user is punished or rewarded, the belief about his or her reputation decreases or increases. These beliefs need to be communicated to the other agents. To enable the agents to communicate with each other, the architecture displayed in Figure 1 is to be extended such that it is able to launch multiple agents simultaneously. This means that the box that represents a single agent in Figure 1 is actually replaced by a framework that can launch multiple instances of agents, which all write the relevant events into one log file.

IV. Behaviour Recognition Engine

There are certain types of behaviour that the users of CTM are not allowed to perform. These behaviours (e.g., following other users, or insulting them) are considered as norm violations, while other types of behaviour (e.g., helping or greeting other users) are considered as norm fulfilling behaviour. To be able to enforce the desired types of behaviour, the normative agents need to recognise the different types of behaviour. However, these behaviours do not map in a one-to-one manner to the raw user actions as received from the SmartFox Server. Examples of these raw user actions are "user A moves to point P" or "user B sends a public message M". Hence, a mechanism is needed to transform the raw user actions into high-level behaviours. For this purpose, the behaviour recognition engine has been developed, which consists of different modules that are responsible for recognising different user behaviours. In the current version of the system, two of such modules have been implemented, namely a Follow Detection Module to recognise users' following behaviour, and a Pattern Matching Module to recognise behaviours based on free text entries, such as insulting and helping. These modules are described in Section 4.A and 4.B, respectively.

A. Follow Detection Module

In the CTM environment, following is considered a norm violation, because when some user A follows user B around (i.e., keeps on copying the movements of user B, no matter where this user is going), this can be perceived as very annoying. Following can also be seen as a form of chasing, a common type of behaviour performed by agents (often referred to as Non-Person Characters) in computer games. In [6] a number of algorithms are described that implement chasing behaviour. However, chasing is a bit different from chase detection, which is the purpose of the current module. To implement chase (or follow) detection, the system needs to detect whether user A is continuously moving to the same location as user B. Nevertheless, even when user A intends to follow user B, the probability that (s)he moves to the exact same location (in terms of pixels) as user B is very small. Therefore, the follow detection module will instead check whether user A moves into the vicinity of user B. The vicinity is defined as the area of 100 pixels around the user (i.e., its 'private space'). The follow detection module keeps track of all users in the virtual world, and counts the amount of times a user steps into the private space of another user successively. If user A enters the private space of user B more than 3 times *without* moving anywhere else[1], then this is interpreted as an instance of following behaviour (we assume that such a sequence of events will hardly ever occur by chance). In that case, the module reports to the normative agent that user A has performed following behaviour.

B. Pattern Matching Module

The goal of this module is to recognise behaviour that has to do with users' text entries. Currently, these types of behaviours are insulting, greeting, helping, and complimenting. To detect those, a simple pattern matching technique is used to analyse the relevant text entries, as often applied in Chatterbots (i.e., computer programs that simulate intelligent conversations). The first Chatterbot was Eliza [15], who simulated a Rogerian psychotherapist. Another well known Chatterbot is A.L.I.C.E. [17], which has its own development language called AIML (Artificial Intelligent Markup Language). A.L.I.C.E. uses a pattern matching algorithm to match the inputted text against patterns which are stored in AIML files.

By slightly changing this technique, it has been adapted for the purpose of identifying the meaning of text messages. For example, when the user inputs the sentence "Hi", this will be recognised as an instance of greeting. Next, this output is sent to the normative agent, which creates a belief about this. In a similar manner, AIML fragments have been created to identify the behaviours 'insulting', 'helping', and 'complimenting' (see the appendix in [13] for examples). Note that the behaviour of 'helping' is a bit more difficult to detect than the other behaviours, since it involves a sequence of communicative acts (e.g., user A first asks a question, which is quickly followed by a response by user B, and possibly by a 'thank you' from user A).

To implement the pattern matching module, the Java-implemented engine of the Alice Chatterbot, called Chatterbean [17] has been used.

V. Normative Agents

Based on observations about the users' high-level behaviours as produced by the BRE, the normative agents

[1] Note that the time span it which this happens does not matter. All that counts is that there are no other actions in between.

have the task to generate appropriate punishments and/or rewards. To this end, they use a BDI-based approach, which has been implemented in AgentSpeak's Java-based interpreter Jason [3]. The line of reasoning followed by each of the agents is visualised in Figure 3.2 of [13]. The most important steps are described in the following subsections. Due to lack of space, the relevant knowledge rules are presented in pseudo-code (in the form of intuitive if-then statements; cf. [5]); for the complete Jason implementation, see [13].

A. *Belief Generation*

For each observed user behaviour, the agent checks whether it concerns a new 'user join' action. If this is the case, this means that we are dealing with a new user, in which case the user is assigned a new (default) reputation:

Belief Generation User Reputation

```
∀u:user
if        observation_result(user_join(u))
and       not belief(exists(u))
then      belief(user_rep(u, 1))
and       belief(exists(u))
```

Reputations are represented by real numbers in the domain [0,1]. The default reputation is 1, i.e., the agents assign a perfect reputation to new users. In case a different action than a join is observed (i.e., an action of a user that was already known to the agent), then the agent generates a belief that the user performed that particular action:

Belief Generation Action Performed

```
∀u:user, ∀ac:action
if        observation_result(performed(u, ac))
then      belief(performed(u, ac))
```

B. *Action Generation*

Based on its beliefs about users' actions, the agent decides whether these actions violate or fulfil certain norms. If this is the case, then an appropriate action is generated, i.e., a punishment or reward. The main assumption behind this approach is that punishing users for certain actions inhibits these actions in the future, whereas rewarding them reinforces those actions.

When deciding how to respond to a particular user's norm violation of fulfilment action, the agent takes two things into account. First, the *level* w1 to which the action fulfils or violates a norm plays a role. This is represented via the predicates fulfil_norm and violate_norm. For instance, the statement

```
belief(violate_norm(follow, treat_respectfully, 0.7)))
```

indicates that the agent believes that the action of following violates the norm to treat people respectfully with a value of 0.7 (in the domain [0,1]). Secondly, the user's current *reputation* r is taken into account. In the rule for generation of rewards shown below, this reputation (again in the domain [0,1]) is multiplied with the level of the norm fulfilment. The result of this mechanism is that users with a good reputation who violate a norm are punished less severely than users with a bad reputation. Similarly, users with a good reputation who fulfil a norm are rewarded higher than users with a bad reputation. Finally, the product of w1 and r is matched to a certain interval, in order to decide which action is chosen. For instance, for rewards, if this product ends up in the interval [0, 0.5], the user receives a compliment. This information is represented using the predicate reward_applicable_to(...). Likewise, if the product is in the domain [0.5, 0.6], the user receives some extra 'nuggets' (CTM's local currency). The rule to determine rewards is represented as follows:

Reward Action Generation

```
∀n:norm, ∀u:user, ∀ac:action, ∀rew:reward, ∀w1,w2,w3,r:real
if        desire(maintain(n))
and       belief(performed(u, ac))
and       belief(fulfil_norm(ac, n, w1))
and       belief(user_rep(u, r))
and       belief(reward_applicable_to(rew, n, weight(w2, w3)))
and       w2 < w1*r
and       w3 >= w1*r
then      perform(reward(rew, u, ac))
```

Note that the rule to determine punishments is similar, with as only difference that fulfil is replaced by violate, reward is replaced by punishment, and w1*r is replaced by 1-((1-w1)*r).

C. *Belief Update*

After punishing or rewarding a user, the agent updates the reputation of this user. The direction (and the extent to which) the reputation is updated depends on (the level of) the norm violation or fulfilment. For this, the following formula is used:

User Reputation Belief Update

```
∀u:user, ∀ac:action, ∀n:norm, ∀w1,r:real
if        belief(user_rep(u, r))
and       belief(performed(u, ac))
and       belief(fulfil_norm(ac, n, w1))
and       belief(user_rep(u, r+w1*(1-r)))
```

Thus, in case of a norm fulfilment, the user's reputation is increased with a number proportional to the level of the fulfilment. Similarly, for norm violations, r+w1*(1-r) is replaced by r–w1*r.

In addition to this, the agent updates its beliefs about the amount of times certain norms have been violated, and certain sanctions have been performed (not shown here). This information can be used to dynamically change the preferences for certain interventions. For instance, when a particular user keeps on violating one specific norm, the agent can decide to punish her more severely.

D. *Domain-Specific Information*

The rules introduced in the previous subsections have been specified in a generic format. To apply them in a particular context, a number of slots have to be filled in with domain-specific information. For example, the domain-specific sort norm has to be filled with concrete norms for the CTM environment. Similarly, the sorts action, reward and punishment have to be filled with appropriate elements. The elements chosen for our application are shown in Table 1.

TABLE I. DOMAIN-SPECIFIC INFORMATION

Sort	Elements
norm	respect_privacy, treat_respectfully, discuss_respectfully, communicate_openly, uphold_society
action	follow, target_user, outing, insulting, arguing, compliment, greet, help, report_violation
reward	increase_user_rep, compliment, receive_nuggets, receive_energy, receive_exclusive_item
punishment	decrease_user_rep, warning, lose_energy, lose_nuggets, kick_from_game, banned_from_game

In addition to this, some of the available desire and belief bases to be initialised. For instance, initially the desire to maintain all norms shown within Table 1 is assigned to all agents. In addition, the agents receive beliefs about the extent to which actions violate and fulfil norms, and about appropriate punishments and rewards for the different intervals (see Section 5.B). For the exact settings of these parameters, see [13].

Before actually implementing the above rules within the CTM environment, their behaviour has been tested in a number of simulation experiments. For this purpose, the LEADSTO environment has been used [5]. This environment takes specifications in terms of causal-temporal relations (as presented in the previous subsections) as input, and produces simulation runs of agent system behaviour in an intuitive format. Due to space limitations, the results of the simulations are not shown here; however, they can be found in [13]. After the simulation results were evaluated positively by the modellers, the specification was considered ready to be implemented within CTM.

VI. EXPERIMENT

To test the behaviour of the implemented system and its impact on its users, a simple experiment has been conducted. The main goal of this experiment was to check whether the visitors of CTM would interact with the normative agents, and whether the agent would respond in the expected manner. To this end, the normative agent system has been introduced in the virtual world for a period of 24 hours. During this period, the environment was open for all potential users worldwide. The normative agent maintained beliefs about the reputation of all visitors (who were not aware of this), and responded to their actions based on the BDI-model. All actions performed by users and agent were logged using the XML Logger.

During the experiment, a total of 10 users have been active in the world. A fragment of the actions logged during the experiment is shown in Figure 2. In this figure, the horizontal axis represents time. The top part of the figure shows some of the actions performed by the users, followed by some of the actions performed by the agent. As shown, many of the actions listed in Table 1 have been performed.

The bottom part of the figure shows the reputation maintained by the agent for one particular user, named Karmijn. As illustrated in Figure 2, this user shows some remarkable behaviour. In particular, she performed the norm violating action follow for 11 consecutive times. For the first four violations, she received warnings from the agent. For the next six violations, nuggets were taken from her, and the last time she performed the action she was removed from the world. So, apparently she was testing the agent on purpose to see what happened until she was removed from the world. What is even more remarkable is that she started to perform norm fulfilling actions after she had been removed from the game. For instance, she started greeting and complimenting other people. Hence, her reputation, which had decreased dramatically, now increased back to its normal value (even though she was not aware of having a reputation).

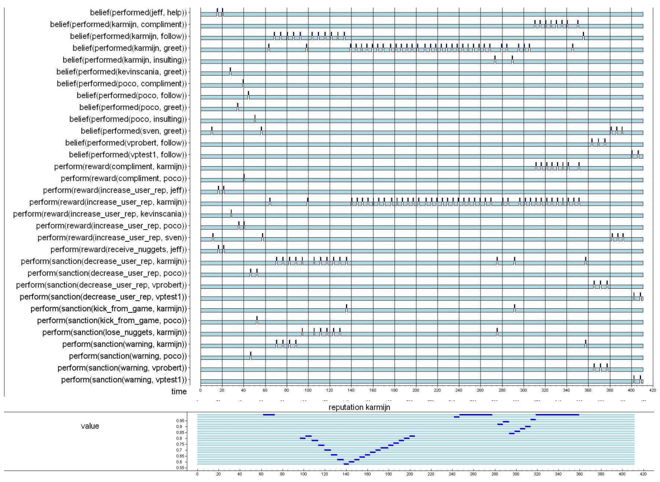

Figure 2. Partial Log of the Experiment.

For the other users, similar observations were made. Although more extensive analyses are needed to demonstrate the successfulness of the system, this is an indication that the system has the potential to adapt users' behaviour by means of punishments and rewards.

A screenshot of the CTM environment, with one of the normative agents in action, is shown in Figure 3. This figure shows the situation of a user who is following another user, in response to which the agent gives him a warning[2].

In addition to observing the results of the logs of the experiment, they have been analysed by means of the TTL Checker [4]. This piece of software takes time-stamped sequences of events (such as the logs of the experiment) as input, and can be used to check automatically if a certain hypothesis holds for these logs. These hypotheses are specified in terms of statements in the TTL language, an extension of order-sorted predicate logic. An example is the following formula, stating that each norm violation is followed (within 2 time points) by a sanction from the agent:

[2] See also http://www.youtube.com/watch?v=7HoF2hwB3L8.

Appropriate Sanctions

$\forall$m:TRACE $\forall$t1:time $\forall$u:user $\forall$ac:action $\forall$n:norm $\forall$w:REAL
state(m, t1) |= belief(performed(u, ac)) &
state(m, t1) |= belief(violate norm(ac, n, w))
$\Rightarrow$ [$\exists$t2:time $\exists$p:punishment
    t1 < t2 < (t1 + 2) &
    state(m, t2) |= perform(punishment(p, u))]

In this formula, state(m,t) |= x denotes that some state property x holds in log file m at time point t. This statement, as well as a number of other hypotheses (see [13]), has been confirmed using the TTL Checker. This can be considered an additional check that the system behaves as it should.

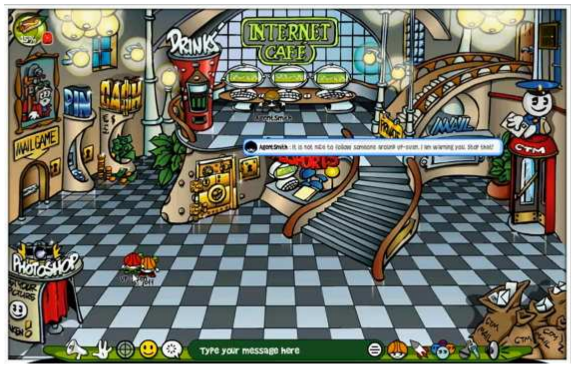
AgentSmith: It is not nice to follow someone around all-over, I am warning you. Stop that!

Figure 3. Screenshot of the CTM Environment.

VII. Discussion

Existing research on preventing the negative effects of cyberbullying via computational methods mainly focuses on peer support by means of affective agents [1, 16]. In contrast, the current paper proposes a system to minimise the amount of occurrences of cyberbullying in the first place. The system is composed of BDI-based normative agents, which are physically present in a virtual society. The agents use a number of intelligent techniques to detect norm violations, and try to reinforce the desired behaviour of the users by means of rewards and punishments. The system has been implemented and tested within Club Time Machine, a virtual environment for children of age 6-12. In a real world experiment, the behaviour of the users of the environment has been logged and analysed by means of a logic-based checking tool. The results show that, on the long term, the system succeeds in reducing the amount of norm violations.

Although these results are promising, more research is needed before the approach is ready to be applied in practice. First of all, the amount of users (and the number of actions performed) involved in the current experiment was too low to derive significant conclusions. Moreover, a number of more detailed questions need to be answered. For instance, do most users indeed learn to fulfil norms over time? And is the mere presence of normative agents in the environment sufficient to enforce norm fulfilling behaviour, or do those agents really have to be active?

To investigate these more sophisticated issues, controlled experiments with potential end users of the system are necessary. One of the goals of these experiments might be to compare a setting with normative agents with a setting without normative agents. A preliminary setup of such experiments is described in [13].

Finally, we intend to extend the Behaviour Recognition Engine with additional modules to recognise other types of norm violations, such as exclusion. As soon as those steps have been taken, the system will be considered sufficiently mature to be applied in a fully open setting.

References

[1] Aylett, R., Paiva, A., Dias, J., Hall, L., and Woods, S. (2009). Affective agents for education against bullying. In: *Affective Information Processing*, Springer Verlag, pp. 75-90, 2009.

[2] Bamford, A. (2004). Cyber-bullying, AHISA Pastoral Care National Conference Melbourne, Australia, Sept. 2004.

[3] Bordini, R., Hübner, J., and Wooldridge, M. (2007). Programming MAS in AgentSpeak Using Jason. John Wiley & Sons, 2007.

[4] Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A., and Treur, J. (2009). Specification and Verification of Dynamics in Agent Models. *IJCIS*, vol. 18, 2009, pp. 167-193.

[5] Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J. (2007). A Language and Environment for Analysis of Dynamics by Simulation. *Int. Journal of AI Tools*, volume 16, issue 3, 2007, pp. 435-464.

[6] Bourg, D.M., Seemann, G. (2004). *AI for Game Developers*. O'Reilly, July 2004.

[7] Cowie, H., Naylor, P., Talamelli, L., Chauhan, P., and Smith, P.K. (2002). Knowledge, use of and attitudes towards peer support: a 2-year follow-up to the prince's trust survey. Journal of Adolescence, vol. 25, issue 5, pp. 453–467, 2002.

[8] Dignum, V., Vázquez-Salceda, J., and Dignum, F. (2004). OMNI: Introduction Social Structure. In: *Norms and Ontologies into Agent Organizations*, 2004.

[9] López y López, F., Luck, M., and d'Inverno, M. (2006). A normative framework for agent-based systems. *Computational and Mathematical Organization Theory*, vol. 12, pp. 227-250, 2006.

[10] Okuyama, F.Y., Bordini, R.H., and da Rocha Costa, A.C. (2008). A Distributed Normative Infrastructure for Situated Multi-Agent Organisations. In: *Proc. of AAMAS'08*, pp. 1501-1504, 2008.

[11] Ptaszynski, M., Dybala, P., Matsuba, T., Masui, F., Rzepka, R., and Araki, K. (2010). Machine Learning and Affect Analysis against Cyberbullying. In: Rzepka, R. (ed.), *Proc. of the Ling. and Cogn. Approaches to Dialog Agents Symposium at AISB'10,* pp, 7-16.

[12] Smith, P.K., Mahdavi, J., Carvalho, M., Fisher, S., Russell, S., and Tippett, N. (2008). Cyberbullying: its nature and impact in secondary school pupils. *J. of Child Psychology and Psychiatry*, vol. 49, issue 4, pp. 376–385, 2008.

[13] Stam, S. (2009). Norm Regulation in Collaborative Virtual Environments by Normative Multi-Agent Systems. M.Sc. Thesis, VU University Amsterdam, 2009. http://hdl.handle.net/1871/19070.

[14] Vázquez-Salceda, J., Aldewereld, H., and Dignum, F. (2004). Implementing Norms in Multiagent Systems. In: Lindemann, G., Denzinger, J., Timm, I.J., and Unland, R. (eds.), *Multiagent System Technologies*, pp. 313-327. Berlin: Springer-Verlag, 2004.

[15] Weizenbaum, J. (1966). Eliza - A Computer Program For the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM*, 1966.

[16] Zwaan, J. van der, Dignum, V., and Jonker, C.M. (2010). Simulating Peer Support for Victims of Cyberbullying. In: *22nd Benelux Conference on Artificial Intelligence, BNAIC'10*, 2010.

[17] http://www.alicebot.org.

[18] http://www.kff.org/entmedia/20021210a-index.cfm.