Applying social norms to implicit negotiation among Non-Player Characters in serious games

Marco Robol
Department of Information Engineering and Computer Science
University of Trento
Trento, Italy
Email: marco.robol@studenti.unitn.it

Paolo Giorgini
Department of Information Engineering and Computer Science
University of Trento
Trento, Italy
Email: paolo.giorgini@unitn.it

Paolo Busetta
Delta Informatica SpA
Trento, Italy
Email: paolo.busetta@deltainformatica.eu

***Abstract*—Believable Non Player Characters (NPCs), i.e. artificial characters simulating rational entities, are a great addition to videogames, no matter if used for entertainment or serious reasons. Especially NPCs that represent people in realistic settings need to show plausible behaviors; to this end, one of main issues to be tackled is coordination with other participants, either other NPCs or human players, when performing everyday tasks such as crossing doors, queuing at an office, picking the first free object up from a set, and so on. Much of this coordination happens silently and is driven by social norms that may vary according to culture and context. In this paper, we propose an approach to represent social norms in autonomous agents and enable implicit coordination driven by observations of others' behavior. Our approach does not use central coordinators or a coordination protocol, but rather let each agent take its own decision so to support more realistic interactions with human players. A software architecture and initial experimental results are presented and discussed.**

I. Introduction

In current videogames, with highly realistic graphics providing an immersive experience, believable NPCs representing humans or other rational entities are an important and largely expected feature. Immersive videogames are increasingly exploited for serious purposes, such as training, where the issue of believability and plausibility of behaviours becomes more relevant than when the objective is pure entertainment.

Smart NPCs represent the evolution of their pre-programmed, context-insensitive older versions. Smart NPCs require sophisticated behavioral models with many facets, including perception elaboration, a decision system, emotions representation, and other aspects driven by cognitive models. Their implementation can be very complex, particularly when they need to interact with other NPCs and, even worse, with human players. Among other abilities, NPCs must use basic coordination mechanisms that are commonly adopted by humans and that could be the result of the application of cultural- and context-sensitive social norms.

This paper reports on the work done within the PRESTO [1] project for NPCs coordination. PRESTO's main objective is the creation of middleware for the support of rational behaviour in videogames. We apply a distributed approach to coordination, called Implicit Negotiation Coordinating Agents (INCA); the idea is to make NPCs aware of social norms and able to interpret and follow them. Specifically, INCA adopts social norms to rule the order of engagement of resources, so that each agent can autonomously reason and implicitly coordinate with others without the need of negotiations. Common examples of situations where INCA is suitable include pedestrians crossing doors or gates, people queuing in front of automatic vending machines, drivers selecting a tollbooth among those at the entrance of a highway, and so on. The ability of silently using social norms is especially important when one of the characters involved is not an NPC but the avatar of a human player.

After a short introduction to PRESTO (Sec. II), we outline our approach (Sec. III), then present some of our current technical choices (Sec. IV) and a few examples (Sec. V). We exploit disjunctive logic to rapidly investigate the representation of social norms in different scenarios; we discuss how this logic-based approach will become the base for implementation in an imperative language, required for practical deployment on commercial systems (Sec. VII). Theoretical underpinning in multi-agent research is discussed in Sec. VIII.

II. Baseline - PRESTO

PRESTO[1] is a suite of development tools and run-time facilities for artificial intelligence in games, focusing in particular on cognitive simulation [1]. PRESTO interfaces a number of game engines, including Unity[2], to compute the perceptions of NPCs and control their behavior by means of autonomous agents developed with PRESTO's own DICE framework.

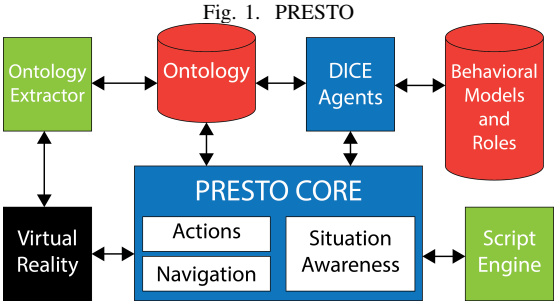
Figure 1 shows a simplified anatomy of PRESTO. Its core provides services such as perception elaboration (Situation Awareness (SA)) and low-level control of the entities in the Virtual Reality (VR). An ontology provides the conceptualization required to make behavioral models and scripts independent of the specificities of a simulation, game or rendering engine [2], [3]. A script engine controls the overall evolution of a simulation, while the DICE multi-agent framework is used for building complex NPC models as an assembly of roles,

[1]Plausible Representation of Emergency Scenarios for Training Operations (PRESTO), a R&D project by Delta Informatica Spa.

[2]unity3d.com, (2016). Unity Technologies [Accessed 26 Feb. 2016].

goals that have to be satisfied by a role and behavioral models implementing the tactics required to achieve goals [1].

Fig. 1. PRESTO



Work is in progress within PRESTO to allow a human player to become the strategic controller of an agent, delegating low level procedures to the latter. This offers the advantages of enabling the construction of cognitive-level GUIs and, internally, of uniformly represent and perceive humans and NPCs.

A. DICE

Delta Infrastructure for Cognition and Emotion (DICE) is a framework to implement autonomous agents capable of controlling NPCs of a VR in real time. It integrates the facilities of PRESTO with AOS' JACK® [4], an agent-oriented development environment built on top of, and integrated with, the Java programming language.

DICE adds a number of facilities to JACK, including a high-level interpreted language and mechanisms to push goals to an agent from the outside, e.g. scripts and GUIs.

DICE takes control of a JACK agent with a scheduler implementing a cyclic process, inspired by the classic OODA loop [5]. Such approach guarantees synchronization with the simulation cycle typical of Computer Graphic (CG) engines. In its first step, called SA Drilling Down, the scheduler cycle processes the perceptions coming from the VR; then, it generates intentions in reaction to perceptions; finally, it gives control to the current main intention (plan) that runs until it decides to yield. During the period in which the plan runs (called "cognitive step") incoming perceptions are queued but not processed by DICE.

Introspection is a general term covering the ability of an agent to reflect upon his own cognitive functions [6]; DICE supports introspection over both goals and intentions. While meta-level reasoning in JACK is limited to choosing, from a pool of applicable plans (intentions), the one to be executed to achieve a goal, DICE supports richer meta-level reasoning thanks to its introspection and attitude manipulation facilities.

III. Methodological approach

INCA is the distributed approach that we propose here to improve the interaction and coordination abilities of NPCs with each other (and potentially with human players). Specifically, INCA handles the question: "what actions, and in which order, should each actor performs to tidily engage shared resources and minimize eventual conflicts?". INCA tackles this coordination problem by reasoning on these concepts: (i) the *actors*, who are the performers of some coordinated actions toward some resources; (ii) the *resources*, as the representations of something that only a limited number of actors at a time can/have to engage; (iii) the *order* of engagement on the various resources that actors have to follow; (iv) the *actions* to be performed by the actors to carry on with the coordination process and finally engage the resources.

INCA focuses on coordination by implicit negotiation, which is very common in people's everyday activities. Each actor computes an order of engagement of resources by itself, without communicating with any of the participants or a central coordinator. This is possible thanks to social norms pre-shared between all the participants, allowing them to generate coherent queues.

Note that agents are autonomous and can therefore decide to break rules; for example, if an agent realizes to be late it could decide to not respect its turn and overcome others actors in the same queue, resulting in a non coordinated behavior.

To apply INCA, we need to identify and give semantics to the actions performed by the actors related to a coordination process. We have identified four actions that actors necessitate and are required to perform in order to coordinate and access resources. Adopting a BDI-inspired terminology, they are called approaching, waiting, engaging and engaged. For example, an actor that wants to enter a room and needs to pass through a narrow door, first gets close to that door (approaching), then, if there are others attempting to pass, waits until those who arrived earlier are gone (waiting), then walks toward the door (engaging) and finally crosses it (engaged).

The agent in control of an NPC reasons on the information it has about the current coordination context, which includes other NPCs involved in the coordination and their current actions, and tries to compute a solution that represents an order of engagement by applying context-specific social norms.

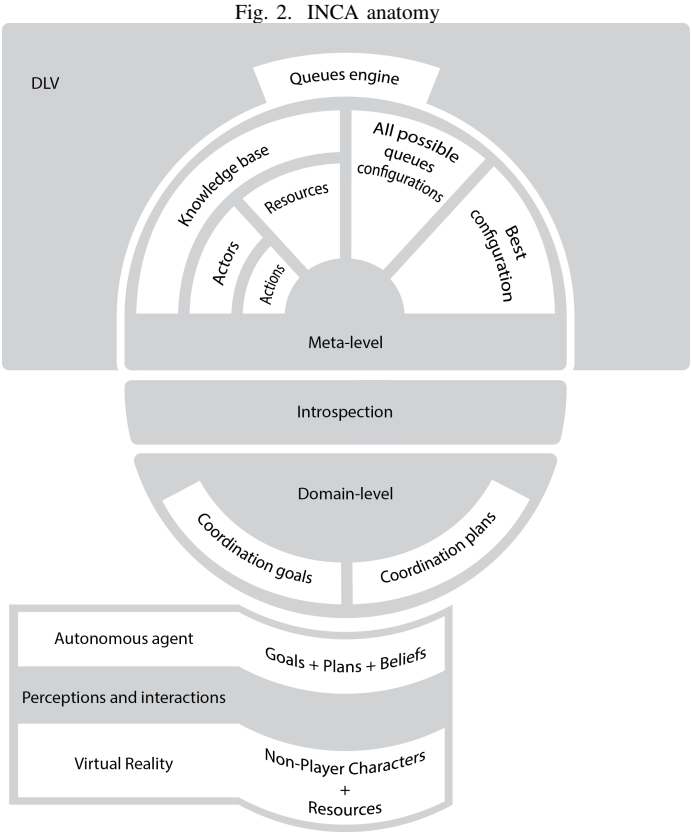
It is worth to stress that information is constrained to what the agent has perceived, and so could be incomplete. Also, the assumption that others are following the same norms is not always valid, thus it is necessary to handle unexpected behaviors (e.g. somebody applying different social norms or even ignoring them because of a strong emotion, such as, panic).

A. Architecture

Figure 2 illustrates the integration of INCA within a single agent controlling a NPC in a VR, thus the represented architecture is repeated for each NPC in a game. In short, social norms for coordination are integrated into each autonomous agent thanks to a meta-level where they are evaluated by an intention deliberation process.

1) Agent domain-level: The domain level is concerned with engaging some specific resources in coordination with others. A coordination desire is achieved by executing intentions performing coordination actions appropriate for the specific

Fig. 2. INCA anatomy



domain of application. Which action, and so which intention, to execute and in which order are decisions taken by the meta-level, which reasons about the coordination independently of the domain.

2) Agent meta-level: Coordination reasoning is performed at meta-level, to keep it separated from the specific domain of application. The meta-level does not make the agent execute any action but operates only on the agent's internal state, controlling intention deliberation thanks to introspection.

To this end, the meta-level evaluates social norms with respect to the current context. These social norms are not built into INCA but provided by the agent when formulating its domain-level coordination desire. Norms may have been selected by a process influenced by different factors, such as a cognitive model that can affect coordination by replacing commonly applied social norms with others that reflect the current emotional state of the agent. For example, if a person is panicking and thus acting irrationally, she does not care to adopt or follow a socially accepted behavior and simply decides to overcame all the other participants and puts herself at the top of the queue.

B. Social norms development and test with DLV

To provide a fast and effective development environment, we opted for a meta-language to represent and reason on social norms.

The meta-language is based on **DLV** [3], a disjunctive logic language that allows Knowledge Representation and Reasoning (KR&R), being one of the best implementation of a Disjunctive Logic Programming (DLP) language [7].In DLP languages, facts describe an initial situations on which the application of disjunctive rules generates different models.

The coordination reasoner takes in **input** a Knowledge Base (KB) containing all the information about the context of coordination, the **actors** and the **resources**, which could be partial because of the limited information available to the agent. A **queue engine** computes all possible combinations of actors in the queues and generates a model for each configuration. Then a set of social norms is applied to filters the solutions, at best, up to one. Finally, an **instructions engine** computes the instructions that each actor should follow.

1) Social norms in DLV: With social norm we intend a filtering criteria of the configurations of actors in the queues, as computed by the queue engine. They are represented, in DLV, as weak constraint rules that rate each configuration giving a weighted penalty to the ones that does not respect them. DLV returns the solution that minimize the number of weighted penalties.

In the example of individuals that want to pass through a door, social norms describe commonly adopted rules to define the order of access; for instance, in normal conditions in a Western country, well-behaving adults respect their order of arrival. In DLV this can be obtained with a weak constraint rule that gives a penalty to each configuration where a person X, who has arrived after Y, is in the queue before the latter: *:∼ next(X,Y), queue(X,C), queue(Y,C), waiting-before-than-for-queue(Y,X,C).* The "next(X,Y)" fact encodes that "Y" will access the resource after "X"; "next" facts are generated for each person to define his queuing order relation with respect to all other people in his own queue. As another example consider the case of multiple queues, where people tend to spread so to minimize the time of access. Such a norm can be expressed by a weak constraint that gives higher penalties to configurations with more people in the same queue: *:∼ next(X, Y).* A long queue is a valid candidate that generates a penalty. Assume that there are two persons and two queues: a configuration in which people are distributed between the queues does not receive any penalty, while a configuration where they are in a single queue gets one penalty.

IV. Toolset support for INCA

INCA has been implemented within the DICE framework described in Sec. II. Its API is currently used by the behavioral models controlling navigation to coordinate the passage through doors, gates and other restrictions along a path. In the following we briefly illustrate the interplay between INCA and BDI domain-specific models.

To allow the automatic recognition of actions performed by other NPCs, INCA exploits the ability offered by PRESTO of dynamically tagging entities in the VR with labels taken from

[3]Dlvsystem.com, (2016). DLVSYSTEM S.r.l. [Accessed 25 Feb. 2016]

its ontology. INCA uses tagging to publish the fact that an NPC is interested in accessing a certain resource and which action it is currently performing. This is required because intention recognition, an innate capability in humans and many animals, is computationally hard if not impossible in a VR given the insufficient level of details corresponding to the clues used by real people.[4]

Initiating coordination on a set of resources (e.g. one or more doors to go from a room to another) can be decided by any behavioral model of the agent. For instance, a plan of the navigation model will start coordination when a door is perceived along the current path. Worth noticing is that the navigation model also exploits PRESTO's dynamic tagging to reduce dependencies from a specific classification of the entities in the VR; a "has crossability" relation can be attached as a so-called PRESTO quality on anything that restricts a path and requires coordination with NPCs arriving from the same or from the opposite direction.

The model invokes INCA to get access to an internal object which contains: (i) a state of the coordination representing the action to perform plus additional parameters, set by INCA itself to drive the domain-level; (ii) the resources to coordinate on and the policy to adopt, which is a set of the social norms valid for the specific domain selected by the model itself at the beginning. This object is passed to a coordination goal that is started by the model itself. A coordination goal is a domain-specific goal (e.g., "door passed") tagged so that the introspection facilities can identify it as controlled by INCA; the goal is achieved when the coordination is concluded (in our example, when the NPC crossed one of the doors).

The BDI plans that are invoked to satisfy the coordination goal are the domain-specific implementations of the generic coordination actions (Sec. III), i.e. approaching, waiting, engaging and engaged. In our navigation example, "approaching" moves the NPC from wherever it is to a certain distance from the door, "waiting" is the (model-chosen) orderly queuing behind the door or crowding in front of it, "engaging" the final move from wherever the NPC is to a distance where it can finally open the door and cross it ("engaged"). INCA first chooses what is the right action to perform then, if a new action is required, uses DICE's introspection API to force the termination of the current action plan and let the normal BDI retry logic to select the appropriate plan given the chosen action (specified in the INCA internal object described above).

INCA exploits the SA step of the DICE scheduler to keep track of what all other NPCs are doing from perception updates; the KB for a coordination goal is built from this data. The SA step is used also to start and monitor an external process executing DLV to compute the resource-access queues and decide which action to take next.

Management of surprises and stalemates arising from conflicting norms is left to future work.

[4]This implies that, in order to achieve NPC / human coordination, players should control avatars at a cognitive level and let their underlying DICE models perform low-level actions requiring coordination, rather than controlling their avatars as puppets as commonly done in videogames.

V. Examples

This section illustrates a few examples of application of INCA. The first ones extend what has been presented in Sec. IV, concerning the crossing of doors during navigation. They show how agents adopt INCA to coordinate access to a single resource, with typical cultural variants captured by social norms. In the last example, concerning a situation of simultaneous emergencies to be tackled with urgency, agents adopt INCA to silently distribute their workload as it would be expected e.g. by a well trained team; this is obtained by modeling the phenomena as multiple resources to be engaged.

A. Crossing doors

PRESTO's navigation subsystem is designed so that agents handle any crossable entity on their path by coordinating access with others and engaging the entity with an appropriate plan (e.g. opening a door, move across it, closing).

To illustrate how this works, consider an NPC "A" that arrives in front of a door. It realizes that the door needs to be crossed and that nobody else is in front, so it approaches the door and engages it. An NPC "B" arrives just after "A" and realizes that "A" was first, so it waits before engaging the door.

In PRESTO, as mentioned above, doors, gates and so on are ontologically tagged as open or closed with the quality "has_crossability". Given this, the navigation model in DICE is able to identify a door on the NPC's current path from the stream of perceptions generated by the PRESTO situation awareness module. As a consequence, the model invokes INCA and gives itself a goal for crossing the door. This goal is tagged as a coordination goal, so that the meta-level is able to recognize it and take its control as discussed in the previous section. Its applicable plans deal with the specific sequences of actions required to implement the current coordination action determined by INCA (initially approaching the door until it is necessary to wait for those in front to cross it, then moving to the distance required to finally open and cross the door). INCA continuously collects and processes data from the agent's memory and incoming perceptions. In the case of the agent B above, at some point INCA decides that it is necessary to wait for A, so, by introspection, the approaching plan is terminated and the state of coordination updated to "waiting", causing the BDI logic to select the related plan to attempt to achieve the coordination goal. Observe that waiting can be implemented in various ways, according to cultural or emotional factors; the simplest option is just stopping the NPC at its current location, without attempting e.g. to queue it behind those in front or getting as close as possible along the shortest path, as required by more realistic modeling. It is left to the agent programmer when developing, selecting or configuring the navigation model for a specific NPC to set it up according to the desired behaviour. Worth noticing is that, as part of its introspection facilities, DICE allows to read the current emotions and to dynamically change the behavioural model concerning a specific capability ("role" in DICE terms),

such as navigation, to select one appropriate for the current state of the character in the simulation.

Let us revisit the case of passage through doors by applying a more sophisticated social norm, based not simply on the arrival order alone but on the socially expected behavior of letting an elderly person pass first (and maybe providing help). To obtain this, the same policy adopted by INCA in the case above is modified to consider these exceptions. Assuming that "B" arrives later as above but represents an elder character, "A" will let it pass first because it determines that this is the correct behavior while "B" passes through the door without waiting because it is also aware of the same rule.

Let us consider a third case of passage through a door, an emergency situation in which the agent-controlled NPCs are in panic. Agents could decide to ignore any social norm by not even invoking INCA and thus attempting to go through the same door. In this case the NPCs move relying only on the physic to handle movement and path conflicts. This recreates a typical situation of panic. Note that this is the same behaviour that is obtained when no coordination mechanisms is applied (even in calm situations), which is one of the reasons for simple simulations to appear not realistic to the observer.

B. Extinguishing fires

PRESTO is being used for emergency training and simulation; a typical problem in this domain is fire management. Let us consider the case of two agent-controlled NPCs, "A" and "B", which could be e.g. firefighters, that need to extinguish a number of fires in a building and need to coordinate in order to decide who tackles which fire.

INCA can be used as in the case of the door above. Each character declares on which resources it wants to coordinate (in this case, a set of fires). INCA automatically manages the allocation of a specific resource among those available when there are many resources and many users applying any desired policy (e.g., the arrival order discussed above), no matter what a "resource" is engaged for.

In our example, if "A" arrives first in the building, its INCA will choose the first perceived fire. When "B" arrives, it perceives "A" engaged with one of the fires and thus, automatically, chooses the first non-engaged one. Once A is done with its chosen fire, it will restart coordination on the set of remaining fires, and so on until all fires are extinguished.

VI. Evaluation - scalability test

We evaluate the scalability of the DLV coordination reasoner, which is the active component of INCA, with respect to the execution time over two factors of complexity: the number of actors and the number of resources. A fixed and simple policy has been adopted for all the tests.

1) Method of execution: The tests have been ran on a machine equipped with an Intel(R) Core(TM) i7-5500U @ 2.4Ghz and 8GB of RAM. A KB is sistematically generated, then the DLV process is ran 5 times, for each combination of the factors of complexity, and the execution times is measured. From the 5 measures, of the same combination, the higher and the lower are removed, then the remaining 3 are averaged.

2) Results interpretation and conclusion: Figure 3 shows the execution time as a function of the number of resources. Figure 4 shows the execution time as a function of the number of actors.

Fig. 3. Execution time with respect to the number of resources

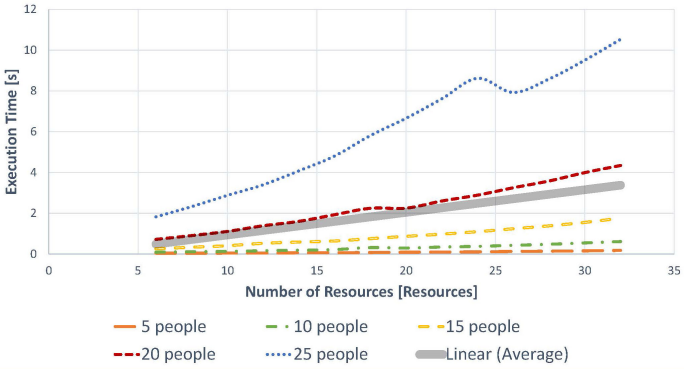


Fig. 4. Execution time with respect to the number of actors

The execution time of a single run of the reasoner increases with the increment of both the number of resources or the number of actors; in the first case the increment is linear, while in the second is exponential. This is due to the increasing number of possible combinations of the actors in the coordination queue.

VII. From disjunctive logic to procedural code

The implementation discussed in the previous sections has the great merit of enabling easy experimentation with different policies. DLV is excellent at expressing constraints in a condensed way, easy to understand and to modify. Further, the data shown in the previous section demonstrate that, for practical purposes, the performance of the current implementation is acceptable at least with up to a few tens of characters. Still, it is not realistic to deploy such an architecture on state-of-the-art personal computers for use by a casual gamer.

We are currently exploring techniques for (automatically or semi-automatically) converting the queue engine and social norms implemented in DLV to a procedural form, suitable for implementation in Java or other common imperative languages. Mimicking the current implementation, a non-optimized procedure would consist of two major steps: com-

puting possible queue configurations and applying weights to order them.

The queue engine, explained in Section III, computes all possible combinations of actors in the queues, which represent alternative configurations to handle the coordination. Procedurally, the same can easily be obtained with nested loops or recursion over the list of actors producing lists representing queue configurations.

At this point INCA needs to apply a policy, a set of social norms, to select one among all the available configurations. Social norms are represented in INCA as DLV weak constraints, which are weighted penalties assigned to configurations given a matching criteria on the characteristics of actors and their order in the configuration. The more a configuration is penalized, the more it is less likely to be the one to be chosen to handle the coordination. An implementation in an imperative language should go through all the queue configurations computed in the previous step and apply all possible matching criteria, incrementing the configuration penalty score every time a match occurs. The winning configuration is the one with the lowest penalty score.

The obvious but important fact to be stressed is that, while in DLV policies are easily represented as weak constraints and new policies can be added at any time, even dynamically (thus supporting learning, for instance), ad-hoc code must be written for each new type of policy to be procedurally supported.

VIII. Related work

This section presents some research works about social norms and the problem of coordination in Multi-Agent Systems (MASs).

1) Social norms and autonomous agents: Wooldridge in [8] defines social norms as an established expected pattern of behaviors, which can be exploited to define coordination mechanisms between agents. Deliberative normative agents are agents that have an explicit knowledge about the enacted norms in a multi-agent environment and can make a choice whether to obey the norms or not[9]. Dignum et al. in [10] propose enhancement of BDI architectures by incorporating social norms, allowing a rich spectrum of social behaviors to be described in a single framework.

2) Coordination approaches in MAS: For Wooldridge in [8] the coordination problem is that of managing interdependencies between the activities of agents.A basic approach to coordinate multiple agents is to restrict their activities in a way which enables them to achieve their goals while not interfering with other agents [11]. Coordination can adopt implicit negotiation, where agents do not explicitly communicate, but negotiation is embedded in a pre-existing context [12].

3) Coordination by social norms: Shoham et al. in [13] ask themselves why not adopt a convention, or, as we would like to think of it, a social law, according to which if each agent obeys the convention, there will be avoided a lot of interactions, creating an implicitly coordinated social behavior without any need for either a central arbiter (to be avoided in MAS) or negotiation (also complex in MAS).

4) Semantic negotiation: Garruzzo et al. in [14] propose a method to form clusters of agents with similar characteristics, or semantically homogeneous, based on semantic negotiation. In comparison to our work it is worth noticing that we do not apply any explicit protocol; rather, an implicit group formation happens by stating interest on a resource and performing actions coherent with what is supposed to be a shared norm.

IX. Conclusion

In this paper, we proposed INCA, a distributed approach to improve the coordination abilities of agent controlled NPCs. INCA integrates social norms into autonomous agents with a meta-level architecture. This allows agents to implicitly coordinate by reasoning and following these social norms. INCA also provides an environment of development and test of these norms based on the DLV language. INCA can be therefore exploited to simulate people implicit coordination mechanisms. An INCA supporting tool-set has been implemented in PRESTO where, in order to evaluate its performance in complex scenarios, we performed scalability tests. As future work, we will work on the implementation of a version of INCA not dependent on DLV.

References

[1] P. Busetta and M. Dragoni, "Composing Cognitive Agents from Behavioural Models in PRESTO," in *Proceedings of the 16th Workshop "From Objects to Agents" (WOA-2015)*, 2015.

[2] M. Dragoni, C. Ghidini, P. Busetta, M. Fruet, and M. Pedrotti, "Using Ontologies For Modeling Virtual Reality Scenarios," in *Proceedings of ESWC 2015*, 2015.

[3] P. Busetta, M. Fruet, P. Consolati, M. Dragoni, and C. Ghidini, "Developing an ontology for autonomous entities in a virtual reality: the PRESTO experience," in *Proceedings of MESAS 2015 workshop*, 2015.

[4] P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas, "Jack intelligent agents-components for intelligent agents in java," *AgentLink News Letter*, vol. 2, no. 1, pp. 2–5, 1999.

[5] J. Tweedale, N. Ichalkaranje, C. Sioutis, B. Jarvis, A. Consoli, and G. Phillips-Wren, "Innovations in multi-agent systems," *Journal of Network and Computer Applications*, vol. 30, no. 3, pp. 1089–1115, 2007.

[6] K. Konolige, "A Computational Theory of Belief Introspection," in *IJCAI*, vol. 85, 1985, pp. 503–508.

[7] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello, "The DLV system for knowledge representation and reasoning," *ACM Trans. Comput. Logic*, vol. 7, no. 3, pp. 499–562, 2006.

[8] M. Wooldridge, *Introduction to MultiAgent Systems*. Hoboken, NJ, USA: Wiley, 2002.

[9] C. Castelfranchi, F. Dignum, C. M. Jonker, and J. Treur, "Deliberate Normative Agents: Principles and Architecture," *Intelligent Agents VI*, vol. LNAI 1757, pp. 364–378, 2000.

[10] F. Dignum, D. Morley, E. Sonenberg, and L. Cavedon, "Towards socially sophisticated BDI agents," in *Proceedings of Fourth International Conference on MultiAgent Systems*, 2000, pp. 111–118.

[11] Y. Shoham and M. Tennenholtz, "On social laws for artificial agent societies: off-line design," *Artificial Intelligence*, vol. 73, pp. 231–252, 1995.

[12] F. Scharpf, "Games Real Actors Could Play: Positive and Negative Coordination in Embedded Negotiations," *Journal of Theoretical Politics*, vol. 6, no. 1, pp. 27–53, 1994.

[13] Y. Shoham and M. Tennenholtz, "On the synthesis of useful social laws for artificial agent societies," in *AAAI-92 Proceedings*, 1992.

[14] S. Garruzzo and D. Rosaci, "Agent clustering based on semantic negotiation," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 3, no. 2, p. 7, 2008.