INTELLIGENT AGENTS FOR SEMANTIC SIMULATED REALITIES
The ISReal Platform

Stefan Nesbigall, Stefan Warwas, Patrick Kapahnke, René Schubotz
Matthias Klusch, Klaus Fischer and Philipp Slusallek
German Research Center for Artificial Intelligence, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany

Keywords: Simulated reality, Multiagent system, Semantic web.

Abstract: Realistic virtual worlds are increasingly used for training, decision making, entertainment, and many other purposes, which require the convincing modeling and animation of virtual characters as well as the faithful behavior of devices in their environment. Intelligent Simulated Realities (ISReal) is a platform for virtual environments that are enriched with a high-level semantic description and populated by intelligent agents using Semantic Web technologies to perceive, understand, and interact with their environment. In this paper, we present the basic architecture of the ISReal platform and show the user interaction in an agent assisted learning scenario.

1 INTRODUCTION

Steadily increasing computational power leverages the use of more advanced techniques from artificial intelligence, computer graphics, and areas that have not been directly associated to virtual worlds, like Semantic Web technology. The development of highly realistic simulated realities is a non-trivial task and a cross-discipline endeavour. Technological development during the recent years mainly focused on the graphical aspect. The behavior of avatars has often been implemented with more or less powerful scripting engines. Instead, the intelligent agent paradigm offers a clean, intuitive, and powerful way of modeling the behavior of intelligent entities in virtual worlds.

Intelligent agents are represented in real-time virtual worlds through avatars (their virtual bodies) and can interact with their environment through sensors and actuators. Sensors cause perceptions that update the agent's beliefs. The agent can reason about its beliefs and plan its actions in order to achieve a given goal. Virtual environments are especially demanding since they are usually dynamic, non-deterministic. To enable agents to interact with their environment a semantic description of this environment is necessary.

3D computer graphic description languages (e.g. X3D, COLLADA) are used in order to describe virtual environments in so called *scene graphs*. These languages specify the objects in the virtual environment by defining their shape, position, orientation, appearance, etc. The X3D/VRML[1] standard (ISO/IEC 19775) has become a unifying (at least conceptual) base, but it lacks when it comes to the high-level descriptions like they are required by intelligent agents. Semantic annotations can be used to link the X3D objects to their semantic description given in a formal semantic description language (e.g. OWL[2]). Furthermore, the functional behavior in the virtual environment can be specified as a semantic service by its input, output, precondition, and effect (IOPE). The formal specification of the virtual world enables the use of Semantic Web technology (reasoning, matchmaking, service composition planning, etc.).

In this paper we introduce the *Intelligent Simulated Reality* (ISReal) platform, which can be used to deploy semantically enabled virtual worlds that are inhabited by intelligent agents. Intelligent agents perceive the semantic annotations of geometric objects and use, beside traditional BDI planning, Semantic Web technology for reasoning and service composition. The platform is based on standards such as X3D for the scene graph, OWL for semantics, and OWL-S for service descriptions.

In the remainder of this paper we provide a detailed overview of the ISReal platform. Section 2 presents the architecture of the whole platform with a

[1] Web3D: http://www.web3d.org/x3d/specifications
[2] W3C: http://www.w3.org/TR/owl-semantics/

Nesbigall S., Warwas S., Kapahnke P., Schubotz R., Klusch M., Fischer K. and Slusallek P. (2010).
INTELLIGENT AGENTS FOR SEMANTIC SIMULATED REALITIES - The ISReal Platform.
In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence - Agents*, pages 72-79
DOI: 10.5220/0002730000720079

focus on the architecture of the agents that can be deployed. In Section 3 we use an agent assisted learning scenario, in which an user has to operate a virtual machine, to demonstrate the interaction of the different components. The user in this scenario can explore the world and interact with agents, e.g. assign tasks like ”Open the door!” or ”Show me what to do in order to get the machine running again.” Finally, Section 4 points out related work and Section 5 concludes the paper.

2 ISREAL SYSTEM ARCHITECTURE

The ISReal platform provides an open and extensible framework for real-time simulated realities. The kind of scenarios that can be deployed cover a wide range of applications such as demonstrators, decision support systems, and virtual training environments. For this purpose, the ISReal architecture has to meet requirements such as (i) scalability in the size and complexity of the simulated worlds, (ii) modularity and exchangeability of the different simulation components, (iii) extensibility (customizability) regarding the supported domains and application scenarios, and (iv) allow a highly realistic simulation of the world regarding geometry, physical properties, and behavior of the entities in a scene. To make our platform as modular as possible, we base our work on standards where possible (e.g. OWL, X3D). One central conceptual building block of our platform is the concept of *semantic objects* which unifies the geometrical shape, semantical properties, and functionality of objects in virtual worlds. The semantic properties are defined with ontologies and the functionality of the objects is described and implemented as semantic services. Services play a key role in our platform. We distinguish between *object services* which are offered by semantic objects in the virtual world, and *platform services* which are offered by the ISReal platform.

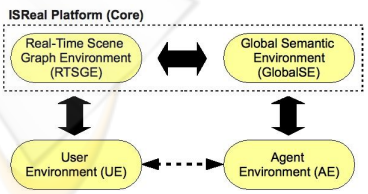

Figure 1: Top-level view of the ISReal architecture.

As depicted in Figure 1, the ISReal platform consists of two core components. The *Real-Time Scene Graph Environment* (RTSGE) manages (i) the geometric representation of the objects in a 3D environment (scene graph), (ii) their semantic meta data, (iii) their physical properties, and (iv) their animations. The *Global Semantic Environment* (GlobalSE) maintains the global semantic high-level representation of the scene and provides implementations of the semantic services to interact with the RTSGE. The RTSGE and the GlobalSE maintain what we call the *Semantic World Model*. Based on these two core components of the ISReal platform, we can connect further modules. The *Agent Environment* (AE) is responsible for the realistic behavior of intelligent entities in the virtual world. It provides the agents (i) with perceptions from the RTSGE, (ii) a semantic knowledge base, (iii) semantic reasoning, and (iv) enables them to invoke object services. A further module is the user environment (UE), which provides the interface for user interactions. For example, the UE also covers the renderer for the scene. In the remainder of this section we provide a detailed overview of the different components of the ISReal platform. We focus on the internal architecture of the agents that can be deployed on the ISReal platform. For this purpose, we first introduce in Section 2.1 the RTSGE and in Section 2.2 the GlobalSE. Both components build the core of the ISReal platform and manage the semantic world model. Section 2.3 is the main part and focuses on the agent architecture. The UE is not covered in this section, but we provide some more information in the example in Section 3.

2.1 Real-time Scene Graph Environment

The RTSGE maintains the geometrical representation and offers an interface for external components to manipulate the virtual environment. We base the scene graph on the X3D standard and use the X3D *Scene Access Interface*[3] (SAI) standard as interface for accessing it. The pure geometric description is enriched by semantic annotations, which is attached directly to the geometric object it belongs to (through X3D metadata objects). The RTSGE communicates with a variety of different services, all of which have a semantic description stored in the GlobalSE (see Section 2.2) and operate directly on the scene graph through SAI.

Semantic objects are a central concept of the ISReal platform. Every semantic object is annotated by (i) the URI that refers to the individual representing a high-level description for this object, (ii) the URIs that refer to the most specific concepts this 3D object belongs to, (iii) the URIs to a set of semantic services (*object services*), describing the useability of

[3]Web3D: http://www.web3d.org/x3d/specifications

the object for a user or an agent. Beside the semantic properties, a semantic object also encompasses (i) geometric information, (ii) animations, and (iii) physical properties like the condition of its surface. These properties are maintained by the RTSGE. Every object interaction an user or agent can perform in the 3D environment is described as an object service in OWL-S[4]. These services are described with IOPE and enable the user or an agent to use Semantic Web technologies to retrieve, plan, or select interaction tasks. The RTSGE has been realized by the *Real-Time Scene Graph* (Rub09).

2.2 Global Semantic Environment

The GlobalSE maintains the high-level description of the scene graph and the semantic services to interact with the 3D environment. It consists of (i) an *Ontology Management System* (OMS) to a) govern the semantic description of the world, b) allow queries in SPARQL[5], a query language to primarily query RDF graphs, and c) semantic reasoning, and (ii) a service registry to maintain semantic services for the interaction with the objects in the 3D environment (object services). With the initialization of the ISReal system, we assume, that the high-level description specified in OWL2-DL ontologies is consistent and accurate to the low-level description specified in the X3D scene graph. The GlobalSE provides an interface to maintain and query the OMS, register/add and read out object services. Figure 2 depicts the core components of the ISReal platform and their interfaces.

The OMS is initialized with OWL2-DL ontologies (global knowledge base *KB*) that are internally stored. It provides maintenance possibilities to add, update, and remove statements from the internal store and if the system shuts down write the store back into an ontology file. To read out the store the OMS provides following types of queries:

a) Object Reasoning. For queries about the concrete objects in the 3D environment, i.e. ABox knowledge retrieval, SPARQL is used. For details we refer to the SPARQL W3C recommendation.

b) Concept Reasoning. To answer questions about the concepts (terminological knowledge) in the 3D environment T-Box reasoning is provided by the OMS in form of a set of fixed methods to answer tasks like global consistency ($KB \models false?$) or class consistency ($C \equiv \bot?$) checks.

c) Relational Reasoning. To find non-obvious relations between different objects, i.e. a set of entities $\{e_1,\ldots,e_n\}$, the OMS can find the smallest tree of the

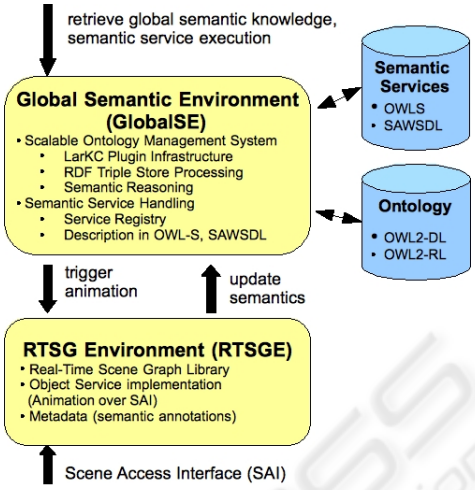

Figure 2: ISReal core components.

RDF graph representing the *KB*, such that it contains all the entities $\{e_1,\ldots,e_n\}$ (Kas09). We provide an example for this kind of query in Section 3.

The implementation uses the LarKC architecture (Fen08). The OMS is implemented as a *LarKC Decider* consisting of different *LarKC Reasoner* plug-ins. As processing plug-ins the OWLIM triple store system (Kir05) and Pellet[6] are used.

2.3 Agent Environment

Intelligent behavior of entities in the ISReal platform is modeled by agents. The agent architecture depends mainly on (i) the properties of the virtual worlds that can be deployed on the ISReal platform, and (ii) the end-user requirements. According to environment properties defined by (Rus03), the environments considered by the ISReal platform are (i) inaccessible, (ii) non-deterministic, (iii) dynamic, and (iv) continuous. The inaccessible property is caused by the fact that an agent only perceives that part of the world which is currently covered by its sensors. Moreover, non-determinism and dynamism are owed to the fact that there is usually more than one agent in the world and the actions performed by these agents can interfere with each other. Furthermore, the number of states that can be reached is not finite, which causes the environment to be continuous. Finally, the agents are acting in a real-time environment, meaning that they have to react in a timely manner. These properties have direct influence on the agent architecture. The end-user of the virtual worlds deployed on the ISReal platform has different possibilities to interact with the

[4]W3C: http://www.w3.org/Submission/OWL-S/

[5]W3C: http://www.w3.org/TR/rdf-sparql-query

[6]http://clarkparsia.com/pellet

agents. He can ask the agent to perform a certain action, assign some declarative goal to it, or can query the agent's local knowledge base.

The ISReal agent architecture is based on the *Belief, Desire, Intension* (BDI) architecture (Rao95) which is well suited for dynamic and real-time environments. Figure 3 depicts an overview of the ISReal agent architecture. We distinguish between the agent core, which encompasses the core functionality provided by an agent execution platform, and the *Local Semantic Environment* (LocalSE), which extends the core with (i) an OWL-based KB (referred to as KB_a), (ii) a reasoner, and (iii) a *Service Composition Planner* (SCP). An agent directly controls its avatar (virtual body of an agent) which is situated in the virtual world. The interface of the agent to its virtual environment is realized by sensors and actuators. Sensors generate perceptions that contain information about semantic objects. Actuators are realized as semantic services that are offered by the agent itself or by the semantic objects the agent can interact with. The perception component is introduced in Section 2.3.1, the information component in Section 2.3.2, and the behavior component in Section 2.3.3.

2.3.1 Perception Component

Sensors provide an agent with perceptions from its current environment. The perceptions in the ISReal platform are caused by the RTSGE which manages the X3D-based scene graph. Since the X3D standard does not specify the kind of sensor that is required by ISReal agents, it is necessary to extend RTSGE with the required functionality. Agents connected to the ISReal platform perceive only semantic objects (see Section 2.1). A perception event contains following information: (i) the object's ID in the scene graph, (ii) the object's individual URI, (iii), the object's concept URI, (iv) the URIs to the object services, and (v) the URIs to the context rules assigned to the object. The perception handling of the agent is done in the following order: (i) receive the perception event coming from the environment, (ii) use the individual URI to get the corresponding ontological facts from the GlobalSE, and (iii) add these facts to the KB_a of the agent's LocalSE (see Figure 3). Additionally, add the URIs of the object services to the respective lists S_a in the LocalSE.

2.3.2 Information Component

To enable agents to process OWL-based information we extended the agent core with the LocalSE. Equivalent to the GlobalSE the LocalSE consists of an OMS that can be queried in the same three ways. Initializing an agent (and with it its LocalSE) the LocalSE has only a partial description of the world representing the knowledge KB_a the agent has in this scenario. The main difficulty is to integrate the LocalSE in a transparent way, so that the agent's internal mechanisms do not break. The information component provides a transparent layer between the agent and its KB_a. It provides functionality for (i) updating and inserting facts into the KB_a, and (ii) reasoning about information in the KB_a. This functionality is used by the behavior component to access the KB_a. For example, we use SPARQL queries in the context condition of BDI plans.

2.3.3 Behavior Component

Classical first-principles planning starts from a given world state and tries to reach an also given goal state by the application of a set of operators. The whole planning process is done off-line, meaning no changes are incorporated during the planning process. BDI-based planners rely on a plan library that provides the agent with plan templates that guide its execution in certain situations (usually defined by relevance conditions and context condition). Because of the plan library, BDI systems are more efficient than classical planners. Their drawback is that they fail as soon as no plan is applicable in the current situations, even if there exists a combination of their plans that achieves a given goal. A further difference is that BDI agents directly execute the actions of a chosen plan template and incorporate incoming events, while classical planners finish the complete planning process before the plan can be executed. One important distinction is the difference between declarative goals (state descriptions) like they are used in traditional planning problems and procedural goals (goal events) of BDI systems that are used to trigger actions (Win02).

The ISReal agent architecture combines the efficiency of BDI-planners and the flexibility of classical planners. An agent's core functionality is implemented as BDI plans and (goal) events. We consider two situations in which a BDI agent benefits from the invocation of a classical planner. The first case occurs when there is no applicable plan in the agent's plan library for the current situation. The agent invokes the classical planner to explore new plans that have not been defined at design-time. The second case occurs when the user assigns a declarative goal to the agent to reach a certain state. The agent can pass this goal to its SCP and gets back a plan consisting of a sequence of services. The agent can either map these services to existing BDI plans or invoke the corresponding object services directly.

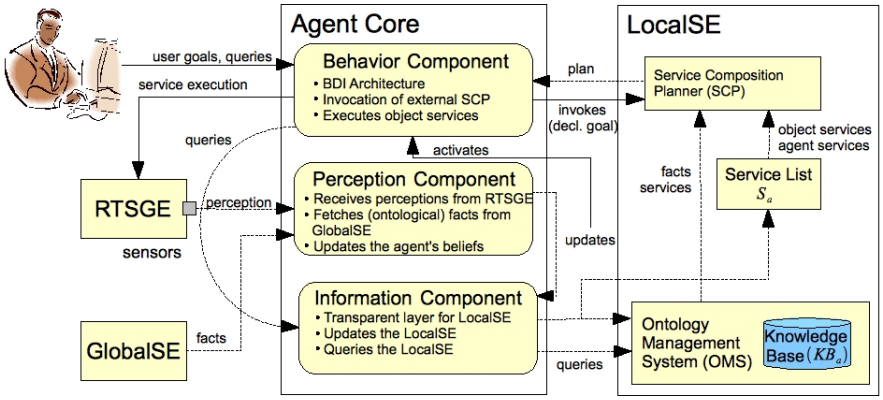

Figure 3: ISReal agent architecture.

Of course, the external planner requires a representation of the BDI plans and (declarative) goals in order to explore new solutions. In ISReal, we therefore specify declarative goals g_d for every goal event e_g in the BDI planner describing the facts an agent wants to achieve when e_g triggers. Furthermore, every BDI plan is described as a semantic service and stored in the service list S_a of his LocalSE (see Figure 3). Whenever a goal event g_e triggers and no BDI plan is applicable, the agent invokes the SCP of his LocalSE. The SCP gets (i) the knowledge base KB_a as first, (ii) the declarative goal g_d of the agent transformed to an OWL2-DL ontology as second ontology, and (iii) all known object services S_a as input. As output the SCP returns a sequence of operations (the plan) and a binding that maps the parameters of each operation to facts in the knowledge base. Using this binding, the agent can execute the services. If the SCP fails to find a plan, then the agent fails to achieve the goal. If the SCP finds a plan, the agent has to execute the operations of that plan. For this purpose, the agent checks for all operations o in the plan whether o is (i) a core service implemented by the agent itself or (ii) an object service that is provided by some object. In the case of a core service, the agent maps the service to a BDI plan and executes the applicable plan. In the case of an object service, the agent fetches the grounding information and invokes the appropriate implementation. Using this combined planning approach the agent is able to find new solutions that are not directly encoded in its plan library. As agent systems we use Jack[7] and Jadex[8]. The SCP has been realized with OWLS-XPlan 2.0, a Semantic Web service composition planner for OWL-S 1.1 services (Klu05).

[7]http://www.agent-software.com/index.html

[8]http://jadex.informatik.uni-hamburg.de

3 EXAMPLE SCENARIO

In the following, we show how the ISReal platform can be used. As introduced in Section 2.2, the user can query the GlobalSE and agents in the virtual world with queries a) for object reasoning (e.g. ”What is this object?”), b) concept reasoning (e.g. ”Are these two objects equivalent?”), c) relational reasoning (e.g. ”What is the relation between the red lamp and the machine?”). Furthermore, the user can execute the functionality of the scene (semantic objects), trigger basic actions of the agent, or formulate a goal (e.g. ”Show me what to do in order to get the machine running again.”, ”Open the door!”) that the agent should achieve.

The scenario considers a new employee whose task is to learn how to use a machine called *Smart Factory* assisted by the ISReal platform. The *Smart Factory* fills pills into cups that are on a transportation belt (see Figure 4). The virtual *Smart Factory* is a simulation of a physical real-world model that is used for demonstrations. In the following the user interacts directly with the machine to turn it on (Section 3.1). After an error occured he asks the agent for more information about the machine (Section 3.2) and finally let the agent show him how to resolve the error (Section 3.3).

In this simple scenario that we use for demonstration purposes, the user can explore a virtual work room, containing the *Smart Factory* and a virtual assistant controlled by an agent. Please note that the transformation from and to natural language is not in the scope of this paper. Therefore, we use only simple template-based transformations. Furthermore, we assume that the user only queries the agent who is familiar with the *Smart Factory* and therefore has an updated local knowledge base that contains all necessary facts.

Figure 4: The Smart Factory use-case.

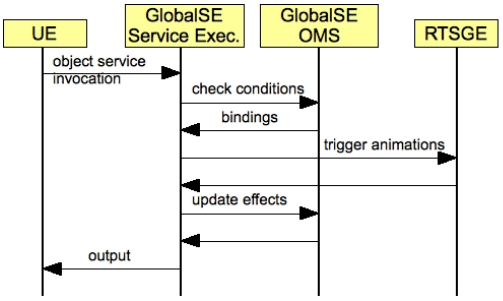

Figure 5: Sequence diagram for object service execution.

3.1 Semantic Object Interaction

The user starts exploring the virtual work room as an interactive immersive environment. He recognizes a control panel at the *Smart Factory*. The user's intention is to put the *Smart Factory* into operation by pressing the button on the control panel. Via the UE he can interact with the *Smart Factory* (as a 3D object) and get all object services the *Smart Factory* provides. These services are *switch_on* and *switch_off*. They are triggered through pressing appropriately labeled 3D buttons on the machine. The user chooses the *switch_on* services and invokes it. Figure 5 depicts a sequence diagram of the whole object service execution. The *switch_on* service expects as input parameter an instance of *Machine* ($?self$). Since $?self$ is the instance of the semantic object that provides this service, the parameter is derived automatically. The precondition is defined as follows: $Machine(?self) \wedge pluggedIn(?self, ?x) \wedge PowerSupply(?x)$. The effect consists of two conditional effects, (a) one for a failed checkup leading to an error state (represented by an active red lamp) and (b) one for a successful checkup leading to the operation state (represented by an active green lamp). (a) is defined as: $consistsOf(?self, ?y) \wedge Magazine(?y) \wedge Empty(?y) \wedge triggers(?y, ?z) \wedge RedLamp(?z)$, having the effect $Active(?self), Active(?z)$. (b) is defined as: $consistsOf(?self, ?y) \wedge Magazine(?y) \wedge Full(?y) \wedge triggers(?y, ?z) \wedge GreenLamp(?z)$, having the effect $Active(?self), Active(?z)$. Let's assume the *Smart Factory* is provided with a power supply but the magazine is empty (which is a fact the user cannot see). After the user turned on the machine the variable binding after checking the conditions is, e.g. $(?self, smartfactory01)$, $(?x, powerSocket01)$, $(?y, magazine02)$, $(?z, redLamp01)$. Based on these bindings, the facts $Active(smartfactory01), Active(redLamp01)$ are derived. The appropriate animations are triggered (given by the grounding information of the service that uses endpoints provided by the RTSGE) and the effect is written to the GlobalSE.

3.2 Information Request

Using the ISReal platform the user was able to put the *Smart Factory* into operation but the machine is in an obvious error state (the signal lamp switches to red). To figure out the problem the user gives following query to the agent assistant: *"What is the relation between the red lamp and the Smart Factory?"* The agent transforms the natural language query into a list of entities and passes it to its LocalSE. The LocalSE handles the query as a relational query (cf. type c) in Section 2.2. By computing the Steiner tree between the entities ('redLamp01', 'smartfactory01') a graph holding the answer is computed: *"The magazine is a part of the Smart Factory that triggers the red lamp."* This answer is produced from the tree given by: *consistOf(smartfactory01, magazine02), triggers(magazine02, redLamp01)*. In order to get more information about the magazine, the user sends a second query to the agent: *"What can you tell me about this magazine?"* After the agent transformed the question into a simple SPARQL query, the query is passed to the agent's LocalSE. As result, the agent returns a set of statements that can be summarized in natural language: *"The name of this object is magazine02. It is a Magazine and Empty. It is part of the smartfactory01 and triggers redLamp01 and greenLamp01."*

3.3 Goal Assignment

After the user gathered information about the relation of the lamps and the machine, he wants to put the machine in its running state (active green light) and asks the agent: *"Show me what to do in order to get the machine running again."* The agent processes the query and handles it as a planning task. Figure 6 shows how the agent processes the query. The user's query is transformed to a declarative goal description: $Active(smartfactory01) \wedge Active(green-$

*Lamp*01*)*. We already introduced the service *switch_on* above. Additionally the agent knows an object service *refill* of the magazine. This service has the precondition *Magazine(*?*self) ∧ Empty(*?*self) ∧ isPartOf(*?*self*, ?*x) ∧ Machine(*?*x) ∧ InActive(*?*x)* and the non-conditional effect *Full(*?*self)*. Please note, that in this example (for the sake of simplicity) we neglect the pills that are actually filled into the magazine. The service *switch_off* checks whether the machine (?*self*) is *Active(*?*self)* and sets it to *InActive(*?*self)*.

Using the SCP the agent determines that it has to use the three services in the sequence: (i) *switch_off*, (ii) *refill*, (iii) *switch_on* with the according variable bindings for the input parameters (i) (?*self*, *smartfactory*01), (ii) (?*self*, *magazine*02), (iii) (?*self*, *smartfactory*01) in order to achive his goal. The first service sets the *Smart Factory* to inactive, which is necessary to fulfill the precondition of the service *refill*. Then the second service can be used to trigger the conditional effect (b) at the service *switch_on* that leads to a state where the goal description is fulfilled (see above). Invoking the plan for every service (cf. loop in Figure 6), the agent either calls the service execution of the GlobalSE, in case of an object service, or triggers the corresponding BDI plan in case of a service describing such a BDI plan. The outputs are used to update the LocalSE of the agent and returned to the user. As visible effect, the agent walks to the machine, unmounts the pill magazine, refills it, mounts it again, and switches the machine back on. In their plans the agents can make use of other services (not discussed here) for navigating and moving in their environment, perform animated actions (like switching a switch on, turning a knob), and the environment will have physical properties and behavior using a physics engine.

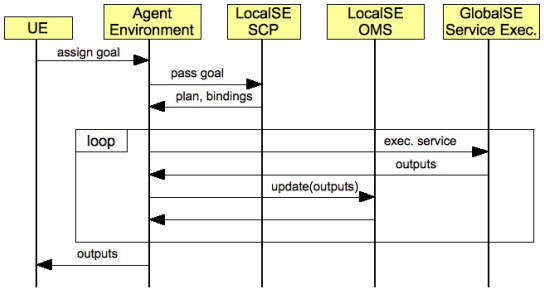

Figure 6: Sequence diagram for goal assignment.

4 RELATED WORK

The central idea of the ISReal platform is to use Semantic Web technology to semantically enrich the pure geometric data of the scene to enable intelligent agents to interact with their environment. However, in a different context semantic annotation of 3D environments has been previously discussed, e.g. in (Pit06), (Bil05), and (Kle07). Kalman et al. (Kal01) proposed the concept of *smart objects* which is a geometrical object enriched with meta information about how to interact with it (e.g. grasp points). Abaci et al. (Aba05) extended smart objects with PDDL data in order to plan with them. Lewis et al. (Lew02) motivates the use of computer game engines in scientific research. For example, (Ork04) presents architectural considerations for real-time agents that have been used in the computer game F.E.A.R.. (Dav06) presents an approach how to connect a BDI agent system to the Unreal engine. (Pan99), (Vos99), and (Ana01) present a multi-agent system for general-purpose intelligent virtual environment applications that consists of three types of conceptually discrete components: worlds, agents, and viewers. (Vos01) presents SimHuman consisting of two basic modules: a 3D visualization engine and an embedded physically based modeling engine. Agents can use features such as path finding, inverse kinematics, and planning to achieve their goals. (Hua02) proposes an approach to 3D agent-based virtual communities in which autonomous agents are participants in VRML-based virtual worlds using the VRML *External Authoring Interface* (EAI). The distributed logic programming language DLP has been extended to support 3D agent-based virtual communities. However, what makes ISReal significantly differ from all existing systems is its integration of virtual worlds, Semantic Web and agent technology into one coherent platform for semantically-enabled agent-assisted 3D simulation of realities.

5 CONCLUSIONS

Semantically enabled simulated realities as considered by this paper have a great potential for commercial applications. Highly realistic prototypes of buildings, production lines, etc. support companies in the early decision making process and help to avoid expensive error corrections. The possibility of training employees on virtual production lines, long before the actual plant has been built, saves time and money.

The realization of a platform for deploying semantically enabled simulated realities is a cross-discipline endeavor and requires input from various research areas such as computer graphics for realistic animations and rendering of a scene, artificial intelligence

for convincing behavior of the entities, and Semantic Web for adding meaning to the purely geometrical objects and describing their functionality. Beside the conceptual integration, requirements such as modularity, scalability, and extensibility are the main drivers for the architecture of the ISReal platform.

In this paper we discussed the basic architecture of the ISReal platform. Key aspects of the future work are on the scalability of core techniques in terms of scene complexity, semantic expressiveness, natural animations, real time performance, and use of many-core hardware.

REFERENCES

Abaci, T., et al.: Planning with Smart Objects. *WSCG SHORT papers proceedings*, UNION Agency - Science Press, pp. 25-28, 2005.

Anastassakis, G., et al.: Multi-agent Systems as Intelligent Virtual Environments. Lecture Notes in Computer Science, Vol.: 2174/2001, pp. 381-395, 2001.

Bilasco, I., et al.: On Indexing of 3D Scenes Using MPEG-7. *Proc. 13th. annual ACM Intl. Conf. on Multimedia*, ACM Press, pp. 471-474, 2005.

Davies, N.P., et al.: BDI for Intelligent Agents in Computer Games. textitProc. of 8th Intl. Conf. on Computer Games: AI and Mobile Systems (CGAMES'06), University of Wolverhampton, 2006.

Fensel, D., et al.: Towards LarKC: a Platform for Web-scale Reasoning, *IEEE Computer Society*, Press Los Alamitos, 2008.

Ghallab, M., et al.: Automated Planning, Theory and Practice. Morgan Kaufmann Publishers Inc., 2004.

Huang, Z., et al.: 3D Agent-based Virtual Communities. *Proc. of the 7th Intl. Conf. on 3D Web technology.*, pp. 137-143, ACM, 2002.

Kasneci, G., et al.: STAR: Steiner Tree Approximation in Relationship-Graphs. *Proc. of 25th IEEE Intl. Conf. on Data Engineering* (ICDE), 2009.

Kallmann, M.: Object Interaction in Real-Time Virtual Environments. PhD thesis, École Polytechnique Fédérale de Lausanne, 2001.

Kiryakov, A., et al.: OWLIM - a Pragmatic Semantic Repository for OWL. *WISE Workshops*, pp. 182-192, 2005.

Klusch, M., et al.: Semantic Web Service Composition Planning with OWLS-Xplan. *1st Intl. AAAI Fall Symposium on Agents and the Semantic Web*, 2005.

Kleinermann, F., et al.: Adding Semantic Annotations, Navigation Paths and Tour Guides for Existing Virtual Environments. *Proc. of the 13th Intl. Conf. on Virtual Systems and Multimedia*, pp. 50-62, Eds., Springer-Verlag, 2007.

Lewis, M., et al.: Game Engines in Scientific Research. *Communcation of the ACM*. Vol. 45, No. 1, pp. 27-31, ACM, 2002.

Orkin, J.: Agent Architecture Considerations for Real-Time Planning in Games. *Proc. of the Artificial Intelligence and Interactive Digital Entertainment Conf. (AIIDE)*, AAAI, 2005.

Panayiotopoulos, T., et al.: An Intelligent Agent Framework in VRML worlds. *Advances in Intelligent Systems: Concepts, Tools and Applications.*, pp. 33-43, 1999.

Pittarello, F., et al.: Semantic Description of 3D Environments: A Proposal Based on Web Standarts. *Proc. of Intl. Web3D Conf.*, ACM Press, 2006.

Rao, S., et al.: BDI Agents: From Theory to Practice. *Proc. of the 1st Intl. Conf. on Multi-Agent Systems*, pp. 312-319, AAAI Press, 1995.

Rubinstein, D., et al.: RTSG: Ray Tracing for X3D via a Flexible Rendering Framework. *Proc. of the 14th Intl. Conf. on 3D Web Technology (Web3D Symposium '09) (to appear)*, 2009.

Russell, S., et al.: Artificial Intelligence: A Modern Approach. 2nd edition. Prentice-Hall, Englewood Cliffs, 2007.

Vosinakis, S., et al.: DIVA: Distributed Intelligent Virtual Agents. University of Salford., pp. 131-134, 1999.

Vosinakis, S., et al.: SimHuman: A platform for real-time virtual agents with planning capabilities. *Lecture Notes in Computer Science*, Vol.: 2190/2001, pp. 210-223, 2001.

Winikoff, M., et al.: Declarative Procedural Goals in Intelligent Agent Systems. *Proc. of the 8th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR2002)*, pp. 470-481, 2002.