Semantic Reflection – Knowledge Based Design of Intelligent Simulation Environments

Marc Erich Latoschik

AI & VR Lab, AI Group, Bielefeld University
marcl@techfak.uni-bielefeld.de,

Abstract. This paper introduces Semantic Reflection (SR), a design paradigm for intelligent applications which represents applications' objects and interfaces on a common knowledge representation layer (KRL). SR provides unified knowledge reflectivity specifically important for complex architectures of novel human-machine interface systems.

1 Introduction

A principle found in intelligent virtual environments [4] is a semantic representation of scene content [2, 3, 5, 7, 8]. Reflecting semantic information on a dedicated KRL has shown to be beneficial for several domains of computational intelligence, from novel – e.g. multimodal – man-machine interactions to virtual agents, or advanced computer games. Semantic models strongly influence recent semantic web efforts and have also gained interest in OOP [6] as enriched representations for object reflection. In the software engineering domain, recent approaches explore the usefulness of ontologies to describe the engineering process of complex systems [1].

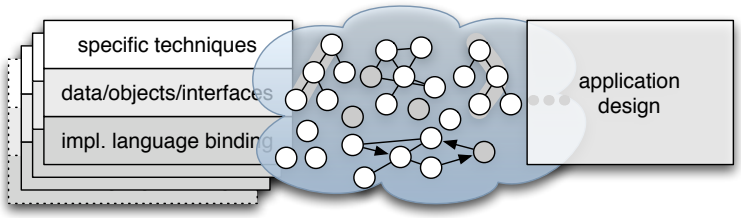

Fig. 1: Semantic Reflection reflects objects, interfaces, and techniques from various modules on a unified knowledge representation layer for high-level application design.

Semantic Reflection (SR) combines and advances these directions. It is a design principle based on a unified semantic description and implementation

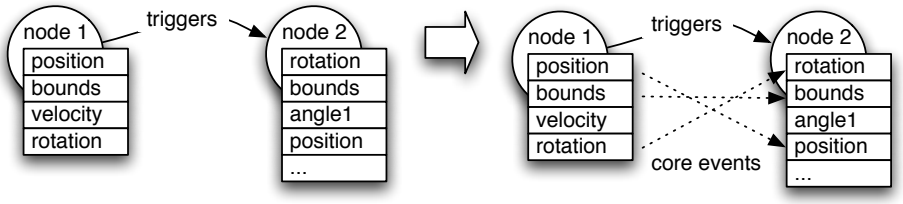

Fig. 2: Semantic representation of application graph technique. Each object is reflected by a semantic net node. A new triggers relation is defined which denotes a required event propagation between the objects. The triggers relation is implemented by the FESN's core event system by establishing core event connections between compatible slots of the connected nodes (see right side).

layer for modular but closely coupled applications which require a built-in knowledge layer support. Integrated into a modular architecture as illustrated in figure 1, SR first establishes a semantic binding to the implementation languages of given modules. Second, it reflects a module's low level design primitives, i.e., the chosen data structures, objects, and interfaces. Third, it reflects the modules' particular techniques like state machines, scene graphs, application graphs etc. To provide Semantic Reflection as a central design feature, a dedicated Functional Extendable Semantic Net (FESN) base formalism provides performance optimizations like hashing and an internal core event system for implementing the procedural semantics for the reflected techniques and interface bindings.

2 Example Module Techniques and Interfaces

Figure 2 illustrates how the base formalism is used to design an application graph, a specific technique for a message system with limited types of events and a fixed chronology of event processing, on the KRL: triggers-relations are directed relations between two nodes. The procedural semantics for the assignment of triggers-relations between nodes is as follows: Core FESN-event connections are established between all compatible slots of connected nodes (see dotted lines in figure 2). This behavior is conveniently implemented using the FESN's functional extendibility by deriving the triggers-relation from the base relation and then redefining its assert method. It is to the developers choice to implement different semantics, e.g., application designers might desire field connections to be established per field and not per object.

There are situations where either the provided modules' techniques are insufficient in terms of a required inter-module data exchange or such dedicated techniques are plainly not provided and modules require direct access to the other modules' interfaces. Hence, modules can export their interface to provide interface reflection on the semantic layer for a high-level definition of function call execution as illustrated in figure 3.

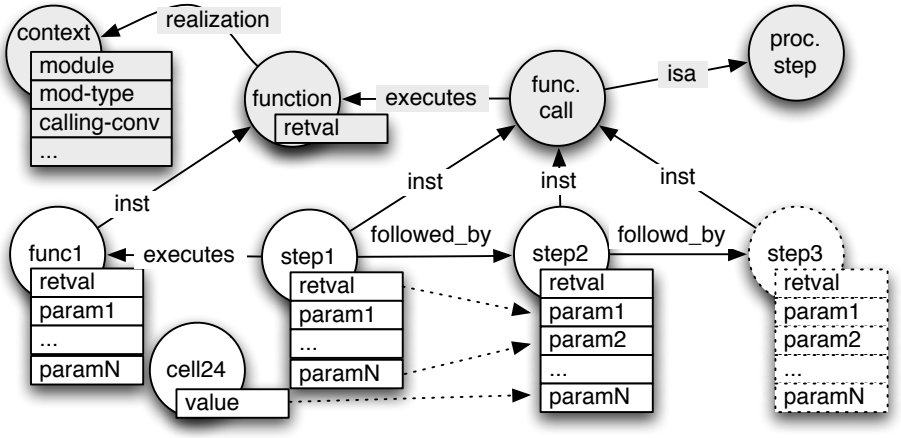

Fig. 3: Basic scheme of function call representation. Functions are lifted to the KRL where they serve as execution targets for processing steps (terminological knowledge represented as grey nodes). The processing steps representing function calls carry a list of slots for their return values and parameters according to the associated functions. The core event system is used to a) define the sources for required parameters as well as b) to finally trigger function execution. Decomposition into function description and processing steps allows for multiple arbitrary processing chains. Independent storage cells are used to insert arbitrary parameters into the call chain between the functions.

3 Conclusion

Semantic reflection has proven to be extremely useful in recent application designs implemented at our lab. It is a novel design paradigm which effectively supports the development of complex but on the other hand extensible and reusable components. As one example, the left pictures in figure 4 are taken during interaction scenes with a virtual agent. The right side illustrates, how semantic reflection of the graphical scene is utilized to reflect the agent's perception as well as the agent's and the user's interaction.

Acknowledgment: This work is partly supported by the DFG under grant Wa 815/2 and the EU project PASION under contract number 27654 in FP6-IST.

References

1. Coral Calero, Francisco Ruiz, and Mario Piattini. *Ontologies for Software Engineering and Technology*. Springer, 2006.
2. E. Kalogerakis, S. Christodoulakis, and N. Moumoutzis. Coupling ontologies with graphics content for knowledge driven visualization. In *Proceedings of the IEEE VR2006*, pages 43–50, 2006.

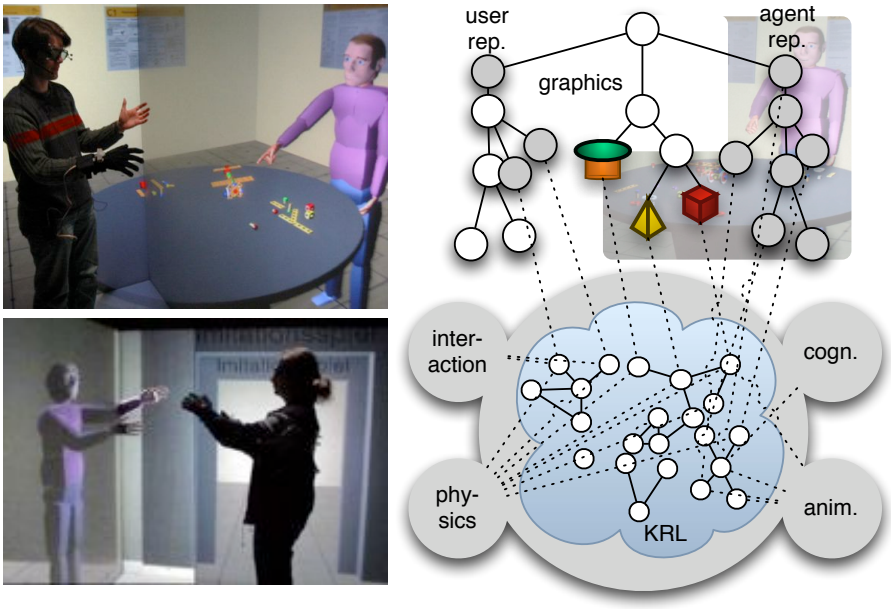

Fig. 4: Left: Interaction with a virtual agent. The agent's perception components automatically access user interactions in the context of the current environment. For example, the agent's vision is implemented as a view sensor monitoring the scene. Using semantic reflection, the agent derives the semantics of what he sees. His higher cognitive processes use this knowledge to further process incoming percepts and to generate appropriate multimodal responses.

3. Marc Erich Latoschik and Malte Schilling. Incorporating VR Databases into AI Knowledge Representations: A Framework for Intelligent Graphics Applications. In *Proceedings of the Sixth International Conference on Computer Graphics and Imaging*, pages 79–84. IASTED, ACTA Press, 2003.
4. Michael Luck and Ruth Aylett. Applying Artificial Intelligence to Virtual Reality: Intelligent Virtual Environments. *Applied Artificial Intelligence*, 14(1):3–32, 2000.
5. Jean-Luc Lugrin and Marc Cavazza. Making Sense of Virtual Environments: Action Representation, Grounding and Common Sense. In *Proceedings of the Intelligent User Interfaces IUI'07*, 2007.
6. José Meseguer and Carolyn Talcott. Semantic models for distributed object reflection. In *ECOOP 2002 - Object-Oriented Programming: 16th European Conference Malaga*, Lecture Notes in Computer Science, pages 1–36. Springer Berlin / Heidelberg, 2002.
7. Stephen Peters and Howie Shrobe. Using semantic networks for knowledge representation in an intelligent environment. In *PerCom '03: 1st Annual IEEE International Conference on Pervasive Computing and Communications*, Ft. Worth, TX, USA, March 2003. IEEE.
8. Michel Soto and Sébastien Allongue. Modeling methods for reusable and interoperable virtual entities in multimedia virtual worlds. *Multimedia Tools Appl.*, 16(1-2):161–177, 2002.