# Towards Semantic Virtual Environments

Karsten Otto

Freie Universität Berlin
Institut für Informatik
otto@inf.fu-berlin.de

**Abstract.** The vision of a global *cyberspace* has inspired a lot of research in the field of Virtual Environments. However, virtual environments today remain island solutions, due to a diversity of incompatible data and protocol designs.
In this paper we introduce the concept of Semantic Virtual Environments (SVE), a scalable approach to this problem, based on the W3C Resource Description Framework. SVE provides a unified machine understandable view on virtual environments, which is suitable both for processing by software agents and presentation to human users.

## 1 Virtual Environments

*Cyberspace. A consensual hallucination experienced daily by billions of legitimate operators, in every nation, by children being taught mathematical concepts... A graphic representation of data abstracted from the banks of every computer in the human system. Unthinkable complexity. Lines of light ranged in the non space of the mind, clusters and constellations of data. Like city lights, receding...*
William Gibson, "Neuromancer"

Virtual environments (VE) today are far from the vision of a global *cyberspace*. They are typically island solutions, custom built for one specific purpose. Many consist of a single monolithic[1] application that tightly integrates networking, data model, and presentation system, possibly even specific input/output devices, each chosen or developed according to the particular application requirements. It is not possible to use one client program to access a different environment, or use an interactive tool in environments that have a different underlying data model. Such tasks require extensive reprogramming or adaptation work. It is the authors opinion that this will remain an issue in the foreseeable future[2].

To illustrate the problem, consider the following example. Three museums in three different cities each have a multi-user virtual environment for virtual exhibitions. Visitors can join these environments with custom client programs to wander around and explore the virtual exhibits. In addition, the first museum provides a museum guide agent. Visitors can point out any exhibit, and the guide provides information about it. The second museum has a tour agent instead, which visitors can ask for a themed museum tour regarding a particular artist, period, or school. The tour agent will lead the visitors through the virtual museum grounds, and point out the respective exhibits.

For further visitor appeal, and true to the cyberspace vision, the museums decide to create a single combined exhibition by linking their exhibitions together. As a consequence, the agents are supposed to integrate exhibits of the other museums into

[1] Alternatively they could use one of the plethora of existing VE toolkits, such as Dive [1].
[2] While there are projects examining various aspects of VE integration [2–4], they do not cover machine understanding for agent support.

their service. The guide agent must be able to answer questions about exhibits in other museums too, and the tour agent must lead the way across virtual environment borders if necessary. This includes the third museum, which did not support any agents before.

The problem is that the three museums use different VE products to realize their respective exhibition. Each has its own proprietary network protocol for coordinating the distributed environment, which includes messages for positioning entities and interacting with them. Furthermore, each virtual exhibition uses a different underlying data model to represent exhibit properties such as title, artist, creation date, and so on.

For the combined exhibition, the client programs must be extended to understand all three kinds of protocols, so visitors can access all three museums. In the same way, both agents must be extended too, so they can detect when visitors point out exhibits they are interested in, as well as move around and point out exhibits themselves. But in addition, they must also be extended to understand all three kinds of data models, so they can retrieve the information about the exhibits to find ones of the requested kind. If a fourth museum joins the combined exhibition, again the clients and both agents must be changed to accommodate yet another protocol and data models. Clearly, this approach does not scale very well. Of course the museums could agree to use a common system for all their exhibitions, but this is unlikely to happen in the presence of existing technology investments.

The important fact here is that all three protocols and data models are used to represent virtually the same things, but in different manners. Although the application domain is consistent – all three VEs are used for virtual exhibitions – the protocol and data model domains are incompatible.

## 2 Semantic Virtual Environments

Semantic Virtual Environment (SVE) is our approach to this problem. We separate all domain semantics from its representation in the data models and protocol messages, and describe it in a neutral machine understandable format. We chose the W3C Resource Description Framework (RDF) [5] for this purpose. It is designed to be used in a highly decentralized manner, appropriate for global information networks, and is well suited for information integration tasks. Through this approach we strive to enable

- uniform access to heterogeneous environments,
- scalable access according to client needs,
- machine understanding for agents, and
- presentation independence for humans.

To illustrate our approach, we revisit the museum example from section 1. The upper half of figure 1 shows a (simplified) SVE description of Museum 1. Here `http://museum1.com` represents the museum *environment* itself; it is marked to be of the type `sve:Environment`, so agents can easily find such environment nodes in the RDF graph. It is also linked to the other two museums; we chose a simple link via an `rdfs:seeAlso` property here, but we could also have used a separate node to describe the link in more detail, in the style of XLink [6]. An agent can use these links to retrieve the SVE descriptions of the other museums, for example to search the whole combined exhibition for a particular exhibit. Also, we can emulate cell-based VE systems, where clients typically use multiple environments at once.

In addition to the inter-environment linking, the SVE description also states that `http://museum1.com` contains an *entity*, the exhibit `http://museum1.com/painting1`. The types and properties of the exhibit are not shown here; an agent

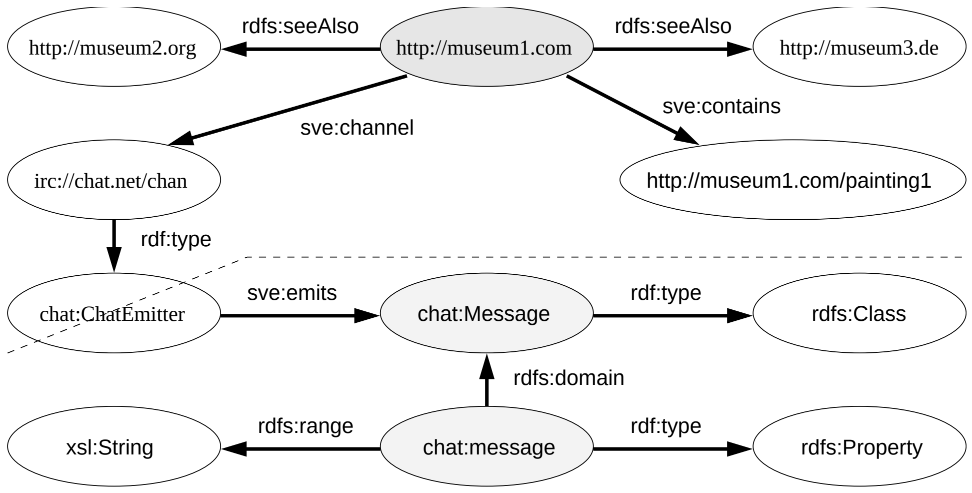

**Fig. 1.** SVE description of a museum (top) and event types (bottom)

could retrieve them via the indicated URL. The entity description will contain a number of `sve:appearance` properties indicating visual representations, such as a 3D mesh or a textual description. Alternatively, these could also be associated with the RDFS [7] type of the entity, whose schema could be retrieved from yet another URL in an additional indirection step. This use of Semantic Web techniques is our way to achieve presentation independence.

Note that the entities contained in an environment are usually not described in a static SVE document. Instead their presence is typically announced dynamically via one of the environments associated channels. A *channel* is the primary means by which a client communicates with a virtual environment. It is some form of network connection, transmitting messages of a particular application protocol. These messages are used to transmit the environments state to its clients, and to distribute change notifications. Typically environments use a form of group communication here, based for example on a UDP multicast group or a central TCP hub server, depending on the particular quality of service requirements of the application protocol. The SVE description associates one or more channels to the environment via the `sve:channel` property. In the example, this is `irc://chat:net/chan`. The description of the channel itself contains all necessary parameters to establish an appropriate network connection, unless this is already covered by the channel URL. More importantly, channels are marked with one or more RDFS types, so agents know what function they serve; in the example the channel is marked `chat:ChatEmitter`.

As our goal is a capturing of semantics, we do not model actual protocol messages themself[3]. Instead, we assume that an SVE client uses some means to convert protocol messages into more abstract *events*. Ideally, an event does not communicate just a simple property or structure change, but indicates that something of importance just happened within the environment. Clients interpret these events, and reproduce any effect this may have on their local copy of the SVEs RDF graph. For example, an event could indicate the arrival of a new visitor, and would be reflected by adding an appropriate `sve:contains` property to the environment.

We represent each event through a small RDF graph, describing the events type and all its parameters[4]. The lower half of figure 1 shows the RDFS schema definition of a `chat:Message` event. Event parameters are represented by properties,

[3] In contrast to other works like GINF [8].
[4] Similar RDF descriptions are made in NEOOM [9], but not for asynchronous events.

in this case the `chat:message` property. It is assigned to the `chat:Message` via `rdfs:domain`, and carrying a `xsl:String` as payload. The type `chat:ChatEmitter` is also part of the schema, related to the message type via `sve:emits`. This indicates that a channel of this type will emit events of the type `chat:Message`. Such information allows agents to select those channels of an SVE whose functionality they need. Figure 2 shows the parts of the SVE description and schema involved in event processing, as well as an example message using the schema.

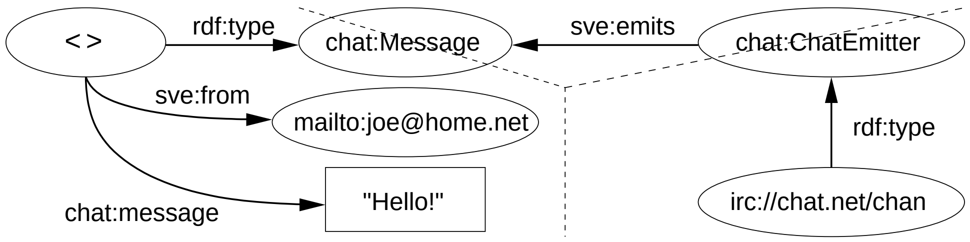

**Fig. 2.** SVE event (left) and typed channel (right)

Modeling events with RDF in this way is useful, since they often need to refer to the environment they occur in. Also, the same mechanisms as for other RDF processing can be applied to them. For example clients may add additional properties during routing, or even merge event graphs to form composite events. However, events differ slightly from usual RDF: The event graphs are separate from the SVE graph, and also from each other, although there is some overlap in the type system. Also, they comprise a closed world, in contrast to the conceptually open RDF world. The model-theoretic foundation of the event concept and its implications are a subject for further study.

## 3 Planned Work and Open Issues

To evaluate our design for SVE, we started the development of SeVEn, aimed to be an extensible framework for the development of portable SVE applications. We plan to continue this development, to validate and improve our current design, and to develop prototype variants to examine the pro and contra of the choices involved.

We also want to analyze more of the existing virtual environment solutions, and integrate them into SeVEn by applying the SVE techniques. We hope to gain valuable insights on potential problems, and to refine and proof our approach.

In addition, we want to investigate the use of event processing mechanisms that can be expressed within RDF itself. These could be dynamically loaded instead of being hard-coded into framework modules or application logic. Rule or script extensions[5] to RDF could be a valuable addition to our framework.

There are a number of open issues regarding RDF modeling, particularly in the areas of inter-environment linking, coordinate systems, and event design. Other issues concern consistency and security in the presence of multiple channels. These are more grave as we cannot change the underlying protocols, and have to work solely with information already available.

Nevertheless, we believe that the research of Semantic Virtual Environments is an important step towards the vision of a global cyberspace.

[5] This was pioneered by cwm [10] and Fabl [11].

## References

1. Frécon, E., Stenius, M.: DIVE: A scalable network architecture for distributed virtual environments. Distributed Systems Engineering Journal (Special Issue on Distributed Virtual Environment) **5** (1998) 91–100 See also `http://www.sics.se/dive`.
2. Zeleznik, B., Holden, L., Capps, M., Abrams, H., Miller, T.: Scene-Graph-As-Bus: Collaboration between heterogeneous stand-alone 3-D graphical applications. Computer Graphics Forum **19** (2000)
3. Capps, M.V., McGregor, D., Brutzman, D.P., Zyda, M.: NPSNET-V: A new beginning for dynamically extensible virtual environments. IEEE Computer Graphics and Applications **20** (2000) 12–15
4. Dachselt, R., Hinz, M., Meiner, K.: Contigra: an XML-based architecture for component-oriented 3D applications. In: Proceeding of the 7th International Conference on 3D Web Technology, ACM Press (2002) 155–163
5. World Wide Web Consortium (W3C): Resource Description Framework (RDF) Model and Syntax Specification. (1999) `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222`.
6. World Wide Web Consortium (W3C): Harvesting RDF Statements from XLinks. (2000) `http://www.w3.org/TR/xlink2rdf`.
7. World Wide Web Consortium (W3C): Resource Description Framework (RDF) Schema Specification 1.0. (2000) `http://www.w3.org/TR/2000/CR-rdf-schema-20000327`.
8. Melnik, S., et al.: Generic Interoperability Framework (1999) `http://www-diglib.stanford.edu/diglib/ginf/WD/ginf-overview/`.
9. Assini, P.: NEOOM: A web and object oriented middleware system (2001) `http://www.nesstar.org/sdk/neoom.pdf`.
10. Berners-Lee, T., Conolly, D.: CWM - Closed World Machine. (2000) `http://www.w3.org/2000/10/swap/doc/cwm.html`.
11. Goad, C.: Describing computation within RDF. In: First Semantic Web Working Symposium, Standford University, California, USA (2001)