Expert Systems with Applications 40 (2013) 1358–1371

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Bridging the HASM: An OWL ontology for modeling the information pathways in haptic interfaces software

Eirini Myrgioti *, Nick Bassiliades, Amalia Miliou

Informatics Department, Aristotle University of Thessaloniki, 541 24 Thessaloniki, Greece

ARTICLE INFO

Keywords:
Haptic devices
Haptic information
Haptic-tactile interfaces
Human–computer interaction
Ontology
OWL
SWRL rules
Software modeling

ABSTRACT

Haptics technology has received enormous attention to enhance human computer interaction. The last decade has witnessed a rapid progress in haptic application software development due to the fact that the underlying technology has become mature and has opened up novel research areas. In an attempt to organize the path between cause and effect we envision a need for a standard for haptic application software modeling. In order for the software to better enhance the tactile information sensation, flow and perception and also make interaction between humans and haptics more efficient and natural, we need a formal representation of the haptics domain. This article proposes the use of HASM, a haptic applications software modeling ontology to formally model the haptics domain in order to be used during the specifications and design phases of developing software applications for haptic interfaces. The presented ontology captures the existing knowledge in haptics domain, using OWL, and defines the pathways that the haptic information follows between the human and the machine haptic system, using SWRL rules. The haptic ontology that has been developed will be used as a basis to design effective user interfaces and assist the development of software modeling for haptic devices. A case study is demonstrating how this haptic ontology can be used to design a software model that analyzes the perception of a haptic property of an object by interacting with a haptic device.

1. Introduction

Haptic technology takes advantage of the sense of touch by applying force, vibration, or motion signals to the user. Current research is focused on tangible interfaces that can be used in a variety of applications reenforcing perception during the interaction between a user and a haptic device (Biggs & Srinivasan, 2002). Haptic information refers to a set of signals that are normally experienced when haptically exploring real, everyday environments. Haptic signals generated when different kind of stimuli are applied and are conveyed to the skin receptors through the skin. The skin receptors transmit tactile signals to the nerve fibers which are responsible to convey the information through multiple pathways to the brain in order to form a perception of an object. The detection and the extraction of the physical parameters of a touched object such as its shape, size, edges, texture, curvature and temperature is one of the main scopes of tactile interfaces in order to form a perception. Physiological and psychophysical knowledge is a crucial part in the design of tangible interfaces (Myrgioti, Miliou, & Chouvardas, 2009).

The development of haptic software systems differs from the development of any traditional software. This work is motivated by the lack of a structured, widely adopted software engineering approach for developing haptic software systems. Many of the existing haptic systems are built as prototypes. Developers still consider haptic software development as an authoring activity rather than an application development to which well-known software engineering practices could apply. Moreover, traditional software engineering methodologies can not be applied as-is, and if applicable they are not precise enough to describe and fit haptic applications (Alamri, Eid, & Saddik, 2006). Haptic hardware and software developers must have a thorough knowledge of the skin physiology, a deep understanding of how the tactile information is conveyed from the haptic device to the human and backwards, and a basic experience in modeling techniques in order to simulate the haptic properties that can be perceived through a haptic device. General object-oriented software engineering approaches such as the Unified Process are not sufficient to model haptic applications as they do not incorporate characteristics unique to the haptic modality such as: haptic rendering, graphic rendering and contact modeling.

So far, the area of ontologies has become attractive for research in Software Engineering because it has been recognized to be useful not only in knowledge-based systems but also in the software development process (Brauer & Lochmann, 2008; Devedzic,

* Corresponding author. Tel.: +30 2310 998407; fax: +30 2310 998419.
E-mail addresses: emyrgiot@csd.auth.gr (E. Myrgioti), nbassili@csd.auth.gr (N. Bassiliades), amiliou@csd.auth.gr (A. Miliou).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.08.053

2002). Before software development, a designer has to have a model of the conceptual structure of the domain i.e. the ontology as well as an understanding of the structure of information describing instances of these concepts and their relationships (Cranefield & Purvis, 1999). Researchers have concluded that ontologies of the software systems application domain, or of its design and construction processes, are of great assistance in avoiding problems and errors at all stages of the software product life cycle (Ruiz & Hilera, 1998). In Wongthongtham, Chang, Dillon, and Sommerville (2005), software engineering concepts, ideas and knowledge along with software development methodologies, tools and techniques have been organized into ontologies which were used as a basis for classifying the concepts in communication and allowing knowledge sharing. However, using formally specified domain knowledge in software design leads to a specific structure of the model (Hruby, 2005). As we need a formal structure of haptics domain, we will illustrate that constructing a haptic ontology will assist the design of software for haptic devices.

The work presented in this paper is based on a previous work (Myrgioti, Chouvardas, Miliou, & Hatalis, 2007) where tactile information ontology has been developed. Specifically the previous ontology focused on the human haptic system. The scope of this paper is: (a) to provide a common and standardized language for sharing and reusing knowledge about haptic domain, (b) to analyze the design of a complete ontological model in this domain that will assist the design of haptic software according to the requirements of an application, (c) to model the different information pathways between the human and the machine haptic system, (d) to identify a basic set of relations between the concepts of the domain (e) to identify a set of rules that provide a dynamic flow of haptic information and (f) to establish the methodology that can be followed in order to built software for haptic interfaces.

This paper is divided in seven sections, which are organized as follows; Section 2 describes the background, the related work and the literature review used in our research. Section 3 introduces the reader to human haptic and machine haptic systems and how the information flow is realized between the two systems. Section 4 describes the methodology that we have followed to design the haptic ontology. Section 5 describes the HASM (Haptic Applications Software Modeling) ontology along with a short description of the rules that have been developed in order to extend the relations between the entities and to use them in haptic software modeling. Section 6 presents the assessment of the haptic ontology with instances and with a software application given a haptic device and a haptic property (roughness). Finally, Section 7 assumes the conclusions and addresses future research directions.

2. Background and related work

2.1. Background

The research direction proposed in this paper builds upon the areas of human haptics, machine haptics and ontologies. Haptic devices are human–computer interfaces that can reproduce the geometry and material characteristics of an object and create haptic stimuli such as Braille, text and graphics that are used in human–computer interaction. Haptic displays are emerging as effective interaction aids for improving the realism of virtual worlds. Being able to touch, feel, and manipulate objects in virtual environments has a large number of exciting applications. The underlying technology, both in terms of electromechanical hardware and computer software, is becoming mature and has opened up novel and interesting research areas (Srinivasan & Basdogan, 1997). Haptic feedback becomes more efficient as computing power increases and proper technology develops. Interacting with a haptic interface, transforms the user's movement through a device to haptic sensory information by properly stimulating the user's haptic and kinesthetic system.

Human haptics includes all aspects of touch and body movement and their application to computer interaction (Hale & Stanney, 2004). The human haptic system consists of the mechanical, sensory, motor and cognitive components of the hand-brain system. In order to develop haptic interfaces that are designed for optimal interactions with the human user, it is necessary to understand the roles played by the mechanical, sensory, motor and cognitive subsystems of the human haptic system. The sensory system includes large numbers of various classes of receptors, nerve endings in the skin, joints, tendons, and muscles as well as areas in the brain that integrate perception.

Appropriate mechanical, thermal or chemical stimuli activate these receptors, causing them to transmit electrical impulses via the afferent neural network to the central nervous system and the brain, which in turn sends commands through the efferent neurons to the muscles for the desired motor action and create the perception of the touched object (Srinivasan, Cutkosky, Howe, & Salisbury, 1999). Tracking and exporting of the physical characteristics of an object such as size, shape, texture, curvature, edges and temperature are the basic processes of haptic interfaces in order to create the sense of perception.

Machine haptics refers to the design, construction, and use of machines to replace or augment human touch; although such machines include autonomous or teleoperated robots, in this work we focus on haptic interfaces to virtual environments. Haptic interfaces are devices composed of mechanical components in physical contact with the human body exchanging information with the human nervous system.

A hardware classification of haptic devices according to the modalities of the skin's sensors which are: static pressure or vibration (mechanical energy), electric field, and thermal flow (temperature difference) can be found in Chouvardas, Miliou, and Hatalis (2008). Another taxonomy of haptic interfaces is according to whether the direct touch and feel of objects contacting the skin is simulated or the interactions are felt through a tool (Srinivasan & Basdogan, 1997). The former is much more difficult since it requires a tactile display capable of distributing forces and torques appropriately over the region of contact between the object and the skin. Machine haptics comprise software techniques and algorithms for haptic applications. Software haptics include various techniques for modeling the virtual objects and their physical characteristics (geometry and material) and also modeling of object kinematics (Barbagli & Salisbury, 2006; Kinashi, Sugisaki, Kanao, Fujisawa, & Miura, 2005; Klatzky & Lederman, 2008; Ho, Basdogan, & Srinivasan, 1999; Salisbury, Conti, & Barbagli, 2004).

In order to write useful application software we need a model of the relevant world (entities, properties and relations). Object-oriented design of software systems depends on an appropriate domain ontology. Specifically, the result of object-oriented analysis is a draft of the domain ontology relevant to the application (Devedzic, 2002). Objects, their attributes and their procedures are more or less mirror aspects of the domain that is relevant to the application (Chandrasekaran, Josephson, & Benjamins, 1999).

2.2. Related work

Ontology-Driven Software Development (ODSD) advocates are using ontologies for capturing knowledge about a software system at development time. So far, ODSD approaches have mainly focused on the unambiguous representation of domain models during the system analysis phase. However, the design and implementation phases can equally benefit from the logical foundations and reasoning facilities provided by the ontology technological space (Brauer & Lochmann, 2008). In Shegogue and Zheng (2005) has been demonstrated that the utility of ontology terms can be enhanced by object-oriented technology, and ontology terms can be integrated into an object-oriented model serving as a basis for the generation of object functions and attributes. By applying object-oriented methodologies and concepts the various domains of a specific ontology can be coordinated into one model. Furthermore, in Evermann and Wand (2005), ontologies have been used to show how basic ontological concepts can be mapped onto an object-oriented equivalent, indicating that object-oriented languages are expressive enough to model real-world application domains while in Bonacin, Baranauskas, and Liu (2004), heuristics and rules have been developed in order to design class diagrams based on ontology charts. By applying them it is possible to produce a first draft of class diagrams useful in a system implementation directed to object-oriented programming paradigm. Moreover, ontologies were used to provide precise execution semantics for modeling software systems and to enhance system architecture design by promoting model reuse (Mokos, Meditskos, Katsaros, Bassiliades, & Vasiliades, 2010).

Haptics modality can be considered as an extra channel of communication that helps software developers design UML models in intuitive and entertaining manner. Haptic systems are complex software systems having the distinguished feature of real-time bidirectional interaction with the human user. The distinction of "input" and "output" is usually very fine and hard to model. Therefore, they require an appropriate software engineering process. In Alamri et al. (2006, 2007), a reference model for haptic software development is presented. The model is designed with the UML and proposes a software modeling technique that comprises modeling elements, notation and methods for haptic software systems. Moreover, in Eid, Alamri, Melhem, and Sa (2008) a prototype haptic-enabled UML CASE tool is presented. That tool allows software engineering developers to intuitively interact and touch the modeling elements of the tool and feel the force feedback.

There have been some efforts to combine ontologies along with human–machine interaction and this has been done for multi-surface interaction and for multimodal environments. Coutaz et.al developed an ontology that shows how the concept of multi-surface interaction can serve as a unifying framework for reasoning about both emerging user interfaces and current interaction techniques such as graphical user interfaces, tangible user interfaces and manipulable user interfaces (Coutaz, Lachenal, & Dupuy-Chess, 2003). Obrenovic et.al introduced an ontology-based approach to present the design of multimodal user interfaces where they have integrated the knowledge and common concepts from different domains of multimodal interaction in a uniform view (Obrenovic, Starcevic, & Devedzic, 2003, 2004). The multimodal ontology is composed of many interconnected ontologies such as the ontologies of human factors and the computing ontologies (Obrenovic et al., 2003). The ontology that has been developed for multimodal interaction includes all aspects of human–machine interaction such as, vision, touch and sound using UML.

In this work we present HASM (Haptic Applications Software Modeling) which is an ontology that captures the entities of the haptic interaction and models the information flow between the human haptic system and the haptic device system. HASM has been built using OWL that aims at representing knowledge about the haptics domain and enabling machines to reason over data in that domain. Moreover, HASM introduces not only the organization of knowledge of haptic interaction in classes, properties and instances but also includes rules that model the different pathways that the haptic information follows depending on the device that is being used.

3. Haptic-tactile information

Generally the term "haptic-tactile information" refers to the information that the human receives through touch. By touching some objects or sensing something cold or hot, we have the haptic sensations. The sensational information (haptic information) when is in touch with an object is discriminated in two categories: (a) *tactile information* defining the information that comes from the skin when is in touch with an object and (b) *kinesthetic information* that comes from the position and the movement of the limbs when forces are applied.

Tactile information is a complex concept because it combines related elements that belong to different systems: the human system and the machine system. The human haptic system includes entities such as the different kind of stimuli, different kind of skin receptors (Iggo, 1984), the groups of nerves that transfer the tactile information (Gardner, Martin, & Jessel, 1991) and the areas of the brain (Gardner & Kandel, 1991) in which the haptic perception happens. The machine haptic system includes: the classes of haptic interfaces and the hardware technologies that are being used to construct haptic devices such as the software methodologies and algorithms that are being used in order to develop applications for haptic systems (Basdogan & Srinivasan, 2002). The properties of these entities and the relations between them describe the procedure of haptic information generation and processing and also haptic information flow from a device to the brain. The processing of tactile signals by the brain leads to the action of perception of the touched object.

However, tactile information is modulated by the channel that is defined by the stimuli-receptors-nerves-brain-perception scheme and different features of a touched object are perceived following different pathways in the above mentioned scheme (Myrgioti et al., 2007). The channel that defines the tactile information is bi-directional and this is the most distinguishing feature of haptic devices when compared to other machine interfaces. A haptic device must be designed to "read and write" to and from for example the human hand (Hayward & Astley, 1996). Modeling the existing knowledge of stimuli-receptors-nerves-brain-perception scheme provides a formal representation of the tactile information domain. Using formally specified domain knowledge in software design leads to a specific structure of the model (Hruby, 2005).

4. Methodology

To create a domain ontology, it is important to find (i) an appropriate set of trees that form its skeleton and that represent ontologically significant categorial distinctions, and (ii) an appropriate set of binary relations (Noy & McGuiness, 2001). Fully structuring domain ontologies in a way that makes them computer-tractable and interoperable, as well as in a way that renders the information that they contain as clear, rigorous, and as unambiguous as possible, requires the use of formal or "upper-level" ontologies (Spear, 2006). The role of an upper-level ontology is to provide the basic categories within which the different tree structures reside, and also to provide a list of binary relations together with axioms that specify their semantics (Bittner & Barry, 2004). There are currently multiple upper-level ontologies such as DOLCE[1] (Gangemi, Guarino, Masolo, Oltramari, & Schneider, 2002; Masolo et al., 2003), SUMO (Suggested Upper Merged Ontology)[2] (Niles & Pease, 2001) and BFO (Basic Formal Ontology)[3] (Grenon, Smith, & Goldberg, 2004). In the context of this paper we consider the BFO reference ontology

[1] http://www.loa-cnr.it/DOLCE.html.
[2] http://suo.ieee.org/.
[3] http://ontology.buffalo.edu/bfo/.

because it is focused on the task of providing a genuine upper ontology that can be used in support of domain ontologies developed for scientific research, for example in biomedicine, even though BFO does not contain physical, chemical, biological or other terms which would properly fall within the special sciences domains.

HASM has been built using the Protege ontology editor,[4] an environment where ontologies could be exported into a variety of formats including RDF, OWL, and XML schema. Specifically, an OWL ontology may include descriptions of classes, properties, restrictions and their instances. Given such an ontology, the OWL formal semantics specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics (Antoniou & van Harmelen, 2008). Our ontology has been developed following the OWL semantics and its structure is based on the OWL implementation of BFO. In this model, the ontology consists of a set of classes organized in the BFO hierarchy to represent the domain's concepts (substances and processes) and a set of properties associated to classes that describe their relations (Noy et al., 2001). HASM reasoning has been carried out using the Pellet 2.0 reasoner[5] which supports the OWL-API, DIG Interface and Jena Interface.

The architecture of the BFO (Basic Formal Ontology) is based on the SNAP-SPAN theory (Grenon, 2003) which allows to talk not only about entities but also about the ontologies through which entities are apprehended. An *entity* refers to everything that exists or occurs in the spatiotemporal world. In BFO, each SNAP ontology represents the entities which fall under the categories of continuant entities (object entities) and each SPAN ontology represents the entities which fall under the categories of occurrent entities (process entities). Both types of ontologies serve as basis for a series of sub-ontologies, each of which can be conceived as a window on a certain portion of reality at a given level of granularity.

Based on our previous work (Myrgioti et al., 2009), where an ontology for human haptics domain was developed with Protege ontology editor in OWL and following the BFO structure, in the paper we extend the ontology by adding: (a) the semantic description of the machine haptics domain, (b) the entities of the human haptics and machine haptics domain, (c) the relations between the entities and some instances and (d) rules that model the dynamic aspects of the flow of information in the various pathways between the human and the machine.

5. HASM ontology

In this section we describe the HASM ontology, the entities of the haptic domain that have been used, the class hierarchy, the properties (relations) between classes, restrictions and finally some instances and rules that model the dynamic aspects of the flow of information in the various pathways between the human and the machine system.

In order to design the HASM, the haptics domain includes all the entities that take part in the processing and the transmission of the haptic information as well as the entities that indicate the manipulation of objects through touch by humans, devices or a combination of them. The haptic interaction can take place in real, virtual or teleoperated environments.

The design of the HASM is based on the two subsystems which are parts of the human-haptic machine interaction (Myrgioti et al., 2009):

- The **Human Haptic System** includes the entities that are related to the sensation and the perception of touch. When a user touches a real or a virtual object, forces are imposed on the skin. The associated sensory information is conveyed from the skin to the brain through nerves and leads to perception. The motor commands issued by the brain activate the muscles and result in hand and arm motion.
- The **Machine Haptic System** includes the technologies of the haptic devices and the interface that is related to the simulation of the sense of touch and the perception of virtual objects. When the user manipulates the end-effector of the haptic device, the position sensors on the device convey its tip position to the computer. The models of objects in the computer calculate in real-time the torque commands to the actuators on the haptic interface, so that appropriate reaction forces are applied on the user, leading to perception of virtual objects (Srinivasan & Basdogan, 1997).

Both systems have sensor mechanisms (receptors-nerves in human system and sensors in machine system), processors (brain in the human and computer in the machine system) and actuators (muscles in human and actuators in machine system). The correlation between the two systems describes the tactile information flow from the user to the haptic device and backwards. The communication between the two systems is developed in the ontology by the relations between the entities of the two systems. The two subsystems and the information flow underlying interactions between users and haptic interfaces are shown in Fig. 1.

5.1. Classes in the HASM ontology

In the previous section, it is emphasized that HASM is designed based on the BFO structure and follows its hierarchy. However, the classification of the entities of the haptics domain is developed based on two criteria:

1. If an entity is an object or a process, or
2. If an entity belongs to a human haptic system or a machine haptic system.

The basic class of HASM is *bfo*:*Entity*. As it is mentioned in the previous section, the class *bfo*:*Entity* has two subclasses: *snap*:*Continuant* for object entities and *span*:*Occurrent* for process entities and each of them is developing a hierarchy of classes. The classes of HASM ontology are built at the last level of the BFO hierarchy structure and are classified in objects and processes.

5.1.1. HASM objects

The HASM Objects hierarchy includes the entities-objects that are parts of the human and machine haptic systems. The main class is bfo:snap:Continuant which has three subclasses: *snap*:*Independent-Continuant*, *snap*:*Dependent-Continuant* and *snap*:*Spatial-Region*.

- *Independent Continuants*: the class *bfo*:*snap*:*Independent-Continuant* includes subclasses of entities which denote objects that their existence does not depend on other entities, have physical limits and spatial position (Grenon et al., 2004). Bfo:snap:IndependentContinuant are divided into: *snap*:*Objects*, *snap*:*Object-Aggregate*, *snap*:*Object-Parts*, *snap*:*Object-Boundary* and *snap*:*Sites*. Each bfo:snap class is divided to human and machine classes. The *snap*:*Objects* class is divided to: *hasm*:*Human-Object* that includes *hasm*:*Brain* and *hasm*:*Hand*, and *hasm*:*Machine-Object* that includes *hasm*:*Actua-tors*, *hasm*:*Sensors* and *hasm*:*Virtual-Objects* classes. The *snap*:*Object-Aggregate* class is divided to: *hasm*:*Human-Object-Aggregate* that includes *hasm*:-*Nerves*, *hasm*:*Receptors* and *hasm*:*Users*, and *hasm*:*Machine-Object-Aggregate* that includes *hasm*:*Haptic-Devices*. *snap*:*ObjectParts* class is divided to: *hasm*:*Human-Object-Parts*

[4] http://protege.stanford.edu/.
[5] http://clarkparsia.com/pellet/.

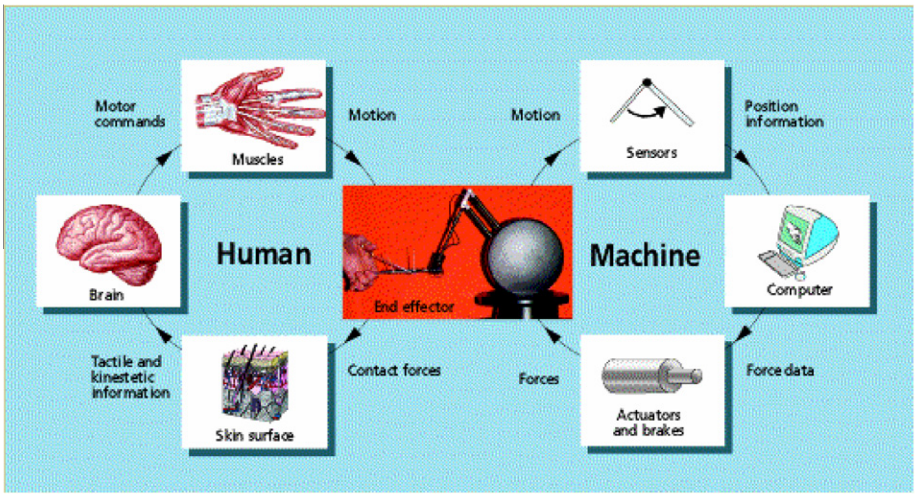


Fig. 1. Haptic interaction between human and machine system (Srinivasan et al., 1999).

class that includes the *BrainAreas* classes that are related to the haptic sense where perception is integrated, and *hasm*:*Machine-Object-Parts* class which includes the *hasm*:*Controllers* class that represent the parts of the haptic machines that drive the haptic devices. The class *snap*:*Object-Boundary* refers to entities that represent surfaces of the objects. That class includes: *hasm*:*Human-Object-Boundary* that contains the classes *hasm*:*BrainCortex* and *hasm*:*Hand-Surface* (such as palm and fingertip that take part in haptic interaction), and *hasm*:*Machine-Object-Boundary* that includes *hasm*:*Object-Surface* class that declares the surface of the virtual objects that can be explored by a haptic device. Finally, the *snap*:*Sites* class includes entities that declare regions where objects are located. For example *hasm*:*Joints* is a subclass of the class *snap*:*Site* because kinesthetic receptors are found in joints. The hierarchy of independent continuants subclasses is presented in Fig. 2.

- *Dependent Continuants*: This class represents entities that endure in time and that inhere in or are born by other entities and has two subclasses: *snap*:*Quality* and *snap*:*Realizable-Entity*. The class *snap*:*Quality* includes classes that represent the characteristics of other entities and the class *snap*:*Realizable-Entity* includes classes that represent disposition, function and role of other entities (human and machine) such as: *hasm*:*Receptors*, *hasm*: *Nerves*, *hasm*:*Sensors*, *hasm*:*Actuators*, *hasm*:*Haptic-Devices and hasm*: *Virtual-Objects*.
- *Spatial Regions*: This is the third of the main subclasses of the class *snap*:*Continuant*. An instance of the class *snap*:*Spatial-Region* is a spatial region, a part of space. All parts of space are spatial regions and only spatial regions are parts of space. *snap*:*Spatial-Region* class is divided to: *hasm*:*Human-Spatial-Region* system that includes *hasm*:*Cortical-Areas-Activation-Point* and *hasm*:*Receptors-Activation-Point*, and *hasm*:*Machine-Spatial-Region* that includes *hasm*:*Interaction-Point class*.

5.1.2. HASM processes

The HASM processes hierarchy includes the entities-processes (occurrents in BFO hierarchy) and their temporal parts that are parts of the human and machine haptic systems and occur at the interaction with a tactile device. The main class is bfo:span:Occurrent which has three subclasses: *Processual-Entity*, *Temporal-Region* and *Spatiotemporal-Region*.

- *Processual Entity*: The class *span*:*Processual-Entity* stand to processes as the class of independent continuants stands to objects and they are entities which exist in time by occurring. They involve object entities and they are dependent on these entities. *Bfo*:*span*:*Processual-Entity* are divided into: *span*:*Process*, *span*:*Process-Aggregate*, *span*:*Process-Part*, *span*:*Process-Boundary* and *span*:*Processual-Context*. Each bfo:span class is divided to human and machine classes. The *span*:*Process* class is divided to: *hasm*: *Human-Process* that includes the *hasm*:*Manual-Exploration* class, *hasm*: *Perception* class and *hasm*:*Stimuli* class, and *hasm*:*Machine-Process* that includes the classes that represent the haptic software techniques such as *hasm*:*Feature-Extraction*, *hasm*:*Geometry-Modeling*, *hasm*:*Haptic-Rendering* and *Kinematics-Modeling*. The class *span*:*Process-Aggregate* has the subclass *hasm*:*Machine-Process-Aggregate* that includes the *hasm*:-*Compu-ter-Haptics* class. Computer haptics represents the sum of processes that describe the haptic software modeling procedure. The *span*:*Process-Part* class has one subclass, *hasm*:*Human-Process-Part* which is divides to: *Haptic Signal Processing by the brain*, *Haptic signal transmission by nerves* and *Haptic signal transmission by receptors*. The class *span*:*Process-Boundary* has one subclass: *hasm*:*User-interaction-with-a-haptic-device*. Finally, *span*:*Processual-Context* class is divided to *hasm*:*Haptic-Feedback* and *hasm*:*Haptic-Signals*. Fig. 3 presents the hierarchy of processes of the human and the machine haptic system from HASM.
- *Temporal Regions*: The class *span*:*Temporal-Regions* represents the regions (intervals) of time during which object and process entities exist. The classes that are subclasses of this class are: *hasm*:*Receptors-Activation*, *hasm*:*Brain-Processing-Duration*, *hasm*:*Cortical-neurons-activation*, *hasm*:*Nerve-fibers-transmission-time*, *hasm*:*Receptors-stimulation*, *hasm*:*Stimulus-time-range*.
- *Spatiotemporal Regions*: The class *span*:*Spatiotemporal-Regions* represents processes at or in which processual entities can be located. The class *hasm*:*Location-of stimulus* is a subclass of this class.

5.2. Relations between the classes of HASM

As mentioned in previous sections, the specification of entities and the organization in classes alone are not enough to adequately capture all of the important information about a given domain. Rather, the relationships obtained amongst the classes in an ontology need to be represented (Spear, 2006). It is also necessary to define and describe the restrictions of classes based on the relationships that members of the class participate in. HASM includes universal, existential, cardinality and hasValue restrictions.

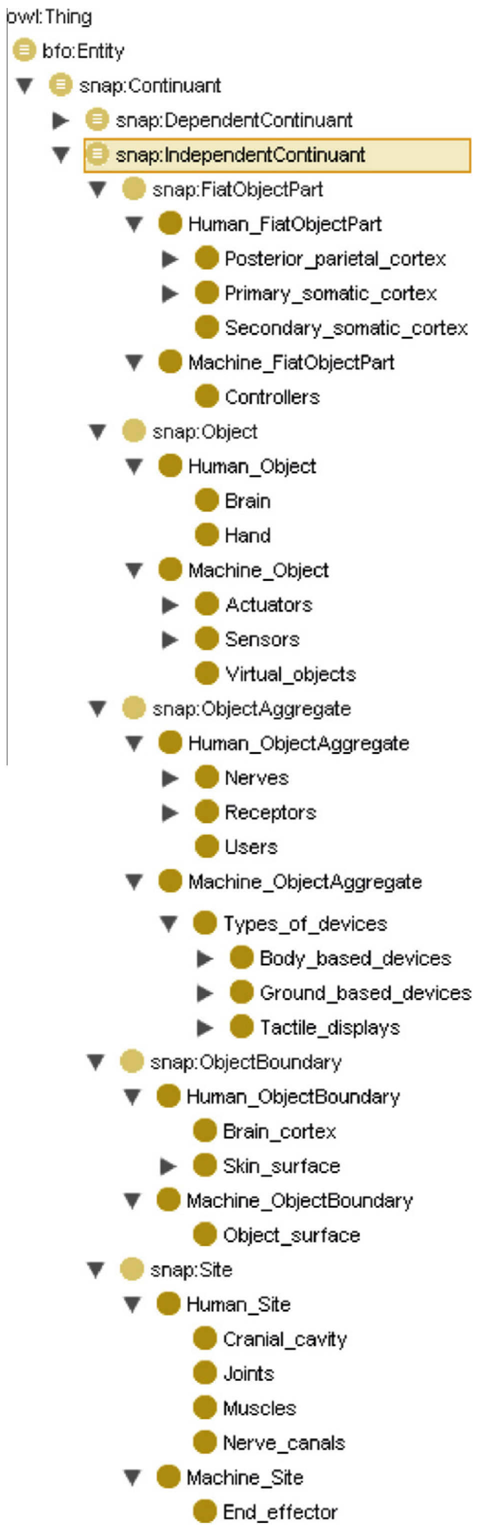


Fig. 2. Subclass hierarchy of the class snap:IndependentContinuant.

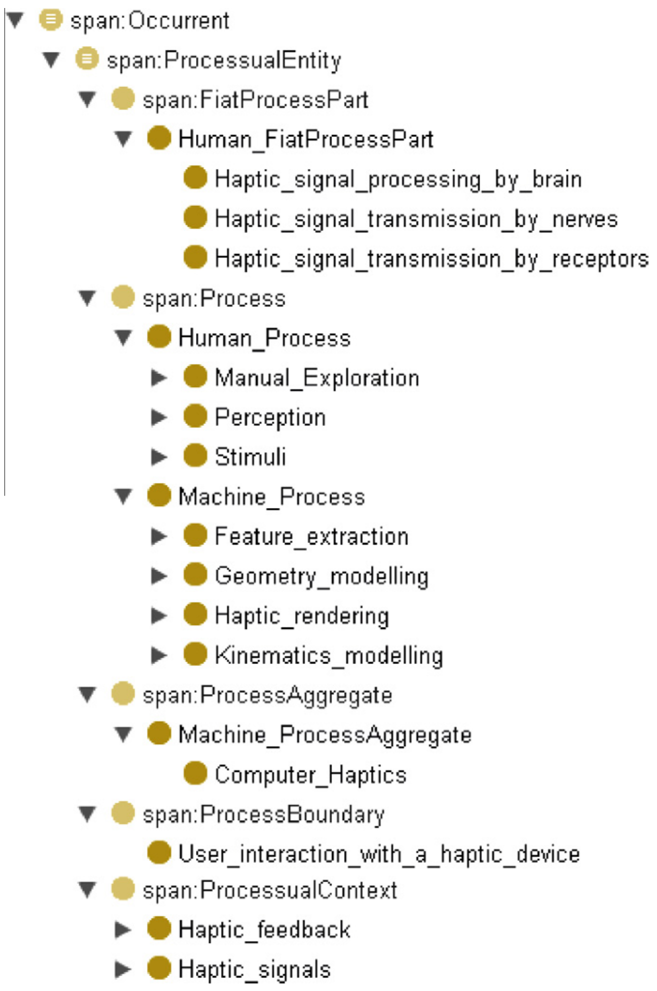


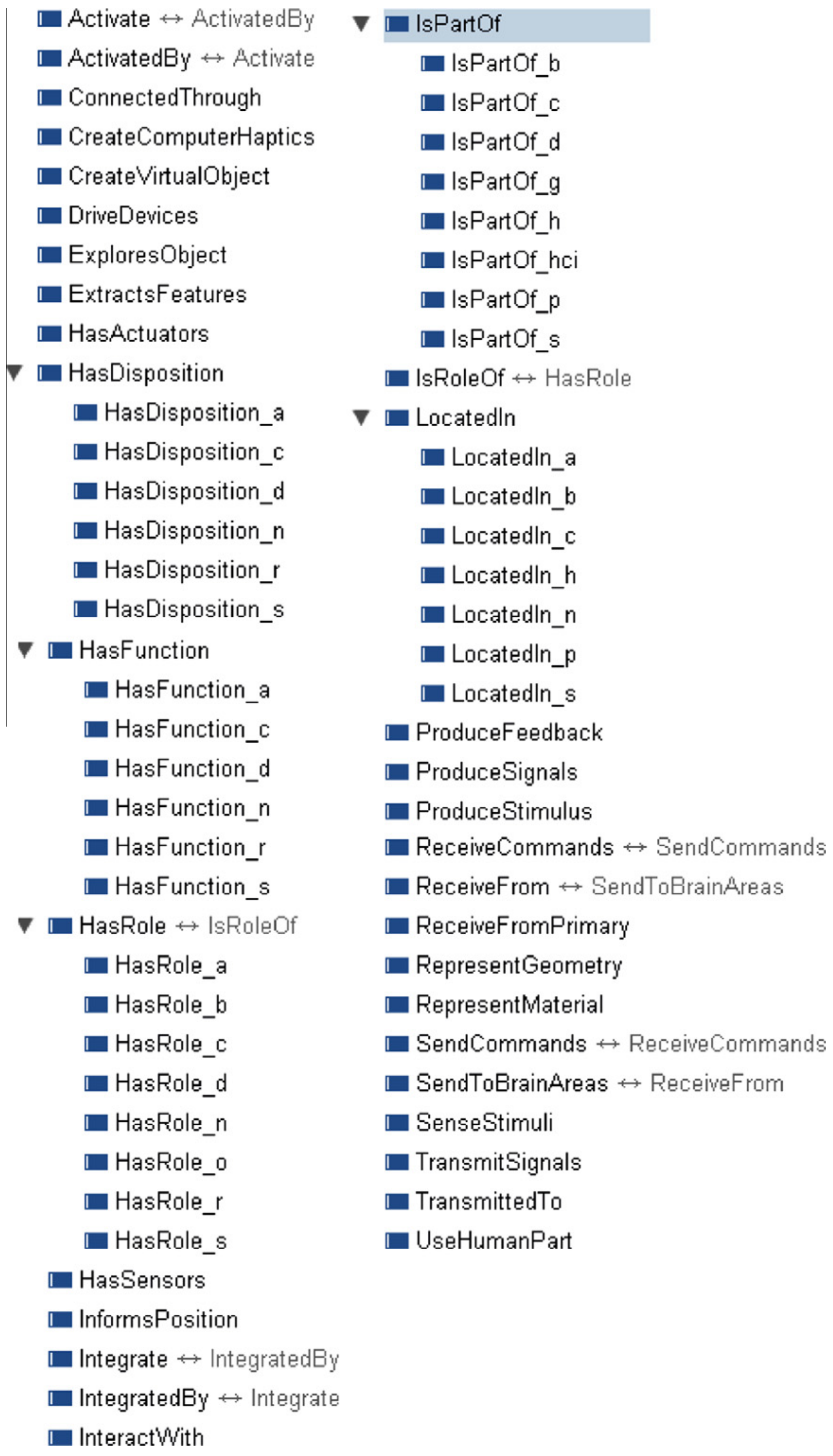
Fig. 3. Subclass hierarchy of the class span:Processual Entity.

HASM allows the definition of correlations between the human and machine systems. In OWL, ontological relations fall into two categories: *object properties* and *datatype properties*.

5.2.1. Object properties

In OWL object properties represent relations between instances. In Fig. 4 object properties of HASM are presented. Some examples of object properties and the defined restrictions based on these properties are:

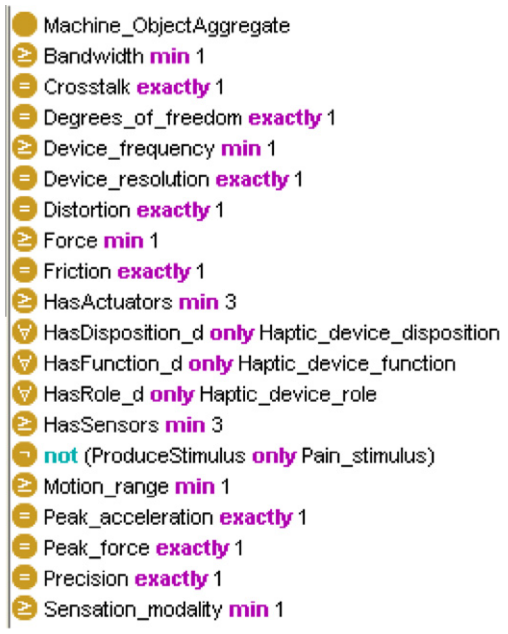
- *activate* (*inverse*: *activatedBy*): relation between Stimuli class (domain) and Receptors class (range), describes which stimuli activate which receptors. For example: **Mechanical-stimulus Activates some (Mechanoreceptors or Proprioceptors or Nociceptors)**.
- *connectedThrough*: relation between Receptors class and Nerves class that describes which skin receptors are connected with the nerves. For example: **RA-I Connected through only Aβ**.
- *hasSensors*: relation between Haptic devices class and Sensors class. For example: **Force-feedback devices hasSensors only Force-torque**.
- *hasActuators*: Each haptic device must have at least three actuators in order to provide high precision in haptic perception. The domain is Types-of-devices class and The range is Actuators class (**hasActuators min 3** – each device must have at least 2 actuators).
- *integrate* (*inverse*: *integratedBy*): relation between brain areas classes and perception classes that describe the haptic characteristic of an object that has been recognized by brain areas. For example, **Area-5 Integrates only Perception-of-shape**.
- *sendToBrainAreas* (*inverse*: *receiveFromBrainAreas*): relation between receptors classes and brain areas classes. Example: **Area 1 receiveFrom only (RA-I or RA-II)**.
- *senseStimuli*: Describes which sensors are activated by which kind of stimuli, for example: **Piezoelectric-sensors senseStimuli some (Mechanical-stimulus or Electrical-stimulus)**.



- Activate ↔ ActivatedBy
- ActivatedBy ↔ Activate
- ConnectedThrough
- CreateComputerHaptics
- CreateVirtualObject
- DriveDevices
- ExploresObject
- ExtractsFeatures
- HasActuators
- HasDisposition
  - HasDisposition_a
  - HasDisposition_c
  - HasDisposition_d
  - HasDisposition_n
  - HasDisposition_r
  - HasDisposition_s
- HasFunction
  - HasFunction_a
  - HasFunction_c
  - HasFunction_d
  - HasFunction_n
  - HasFunction_r
  - HasFunction_s
- HasRole ↔ IsRoleOf
  - HasRole_a
  - HasRole_b
  - HasRole_c
  - HasRole_d
  - HasRole_n
  - HasRole_o
  - HasRole_r
  - HasRole_s
- HasSensors
- InformsPosition
- Integrate ↔ IntegratedBy
- IntegratedBy ↔ Integrate
- InteractWith
- IsPartOf
  - IsPartOf_b
  - IsPartOf_c
  - IsPartOf_d
  - IsPartOf_g
  - IsPartOf_h
  - IsPartOf_hci
  - IsPartOf_p
  - IsPartOf_s
- IsRoleOf ↔ HasRole
- LocatedIn
  - LocatedIn_a
  - LocatedIn_b
  - LocatedIn_c
  - LocatedIn_h
  - LocatedIn_n
  - LocatedIn_p
  - LocatedIn_s
- ProduceFeedback
- ProduceSignals
- ProduceStimulus
- ReceiveCommands ↔ SendCommands
- ReceiveFrom ↔ SendToBrainAreas
- ReceiveFromPrimary
- RepresentGeometry
- RepresentMaterial
- SendCommands ↔ ReceiveCommands
- SendToBrainAreas ↔ ReceiveFrom
- SenseStimuli
- TransmitSignals
- TransmittedTo
- UseHumanPart

Fig. 4. Object properties of HASM.

- *produceFeedback*: relation between haptic devices and haptic feedback classes, for example: **Tactile-displays produceFeedback only Tactile-feedback**.
- *produceStimulus*: relation between haptic devices and stimuli classes, for example: **Vibration displays produceStimulus only (Mechanical-stimulus or Force-stimulus)**.
- *representGeometry*: describes the modeling of virtual objects, relation between Geometry modeling and geometry features classes, for example: **Geometry modeling representGeometry some (Shape or Size or Orientation or Weight-volume)**.
- *representMaterial*: relation between haptic rendering and material features of objects classes. Material features of objects are rendering using haptic rendering techniques and algorithms. For example: **Haptic rendering representMaterial some (Compliance or Temperature-Thermal-quality or Texture or Weight-density)**.
- *extractsFeatures*: Relation between haptic devices and object properties classes and describes which haptic property of a virtual object can be perceived by a haptic device. For example: **Force-feedback devices extractsFeatures only (Shape or Texture or Compliance or Weight-volume)**.



```
Machine_ObjectAggregate
Bandwidth min 1
Crosstalk exactly 1
Degrees_of_freedom exactly 1
Device_frequency min 1
Device_resolution exactly 1
Distortion exactly 1
Force min 1
Friction exactly 1
HasActuators min 3
HasDisposition_d only Haptic_device_disposition
HasFunction_d only Haptic_device_function
HasRole_d only Haptic_device_role
HasSensors min 3
not (ProduceStimulus only Pain_stimulus)
Motion_range min 1
Peak_acceleration exactly 1
Peak_force exactly 1
Precision exactly 1
Sensation_modality min 1
```

Fig. 5. Types-of-devices class definition.

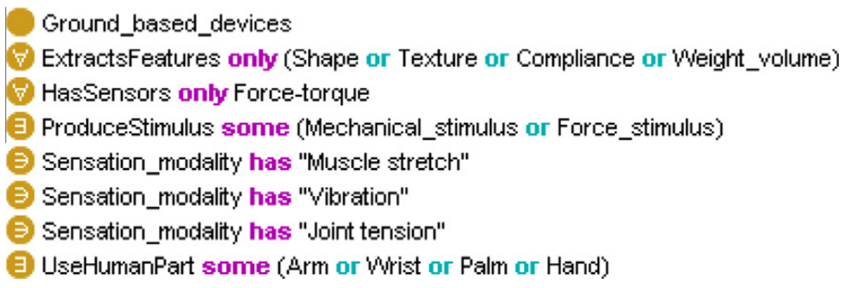
- Properties such as hasFunction, hasRole, hasDisposition, isPartOf and locatedIn have subproperties such as: **hasFunction-a** (Actuators hasFunction-a Actuator-function), **hasRole-n** (Nerves hasRole-n Nerves-role), **hasDisposition-r** (Receptors hasDisposition-r Receptors-disposition), **isPartOf-c** (Haptic-rendering and Kinematics-modeling and Geometry-modeling and Feature-extraction is-Part-Of-c Computer-Haptics), **locatedIn-s** (Mechanoreceptors or Nociceptors or Thermoreceptors locatedIn-s only Skin-surface).

5.2.2. Datatype properties

Datatype properties connect an instance to a datatype value. The property "Involves" can have two values: human and machine and has been created to discriminate human from machine classes and instances (necessary and sufficient condition). Properties such as degrees of freedom, inertia, damping, motion range, dynamic precision, resolution, friction, peak force and acceleration, bandwidth and force are performance measures of haptic devices. The benefits that follow from these measures are numerous, for example:

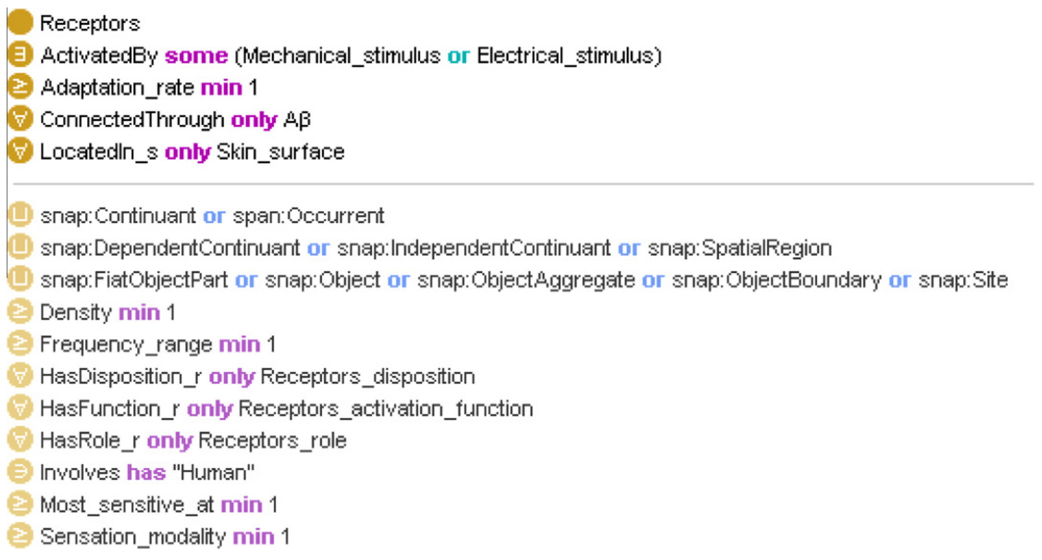
- Device performance and price can be matched in an informed fashion to the tasks they are meant to address.
- Devices can be specified before they are built.
- Devices with different designs can be compared (Hayward & Astley, 1996; Laycock & Day, 2003).

Properties such as *density*, *adaptation rate*, *frequency range*, *receptive field and time range* belong to skin receptors determining their sensitivity to the incoming signals (Hale & Stanney, 2004) and must be take them into account in the construction of a haptic device and the design of haptic interfaces. For example each class of



```
Ground_based_devices
ExtractsFeatures only (Shape or Texture or Compliance or Weight_volume)
HasSensors only Force-torque
ProduceStimulus some (Mechanical_stimulus or Force_stimulus)
Sensation_modality has "Muscle stretch"
Sensation_modality has "Vibration"
Sensation_modality has "Joint tension"
UseHumanPart some (Arm or Wrist or Palm or Hand)
```

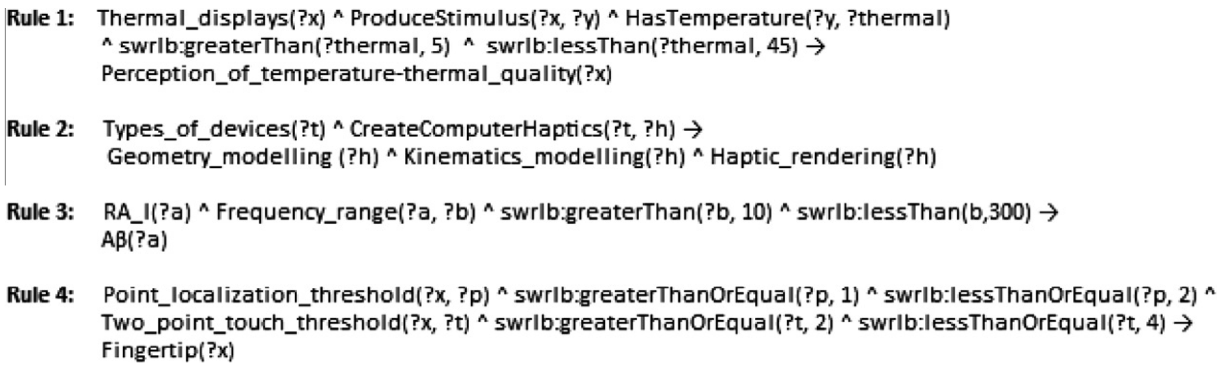
Fig. 6. Force-feedback-devices class definition.



```
Receptors
ActivatedBy some (Mechanical_stimulus or Electrical_stimulus)
Adaptation_rate min 1
ConnectedThrough only Aβ
LocatedIn_s only Skin_surface

snap:Continuant or span:Occurrent
snap:DependentContinuant or snap:IndependentContinuant or snap:SpatialRegion
snap:FiatObjectPart or snap:Object or snap:ObjectAggregate or snap:ObjectBoundary or snap:Site
Density min 1
Frequency_range min 1
HasDisposition_r only Receptors_disposition
HasFunction_r only Receptors_activation_function
HasRole_r only Receptors_role
Involves has "Human"
Most_sensitive_at min 1
Sensation_modality min 1
```

Fig. 7. Mechanoreceptors class definition.



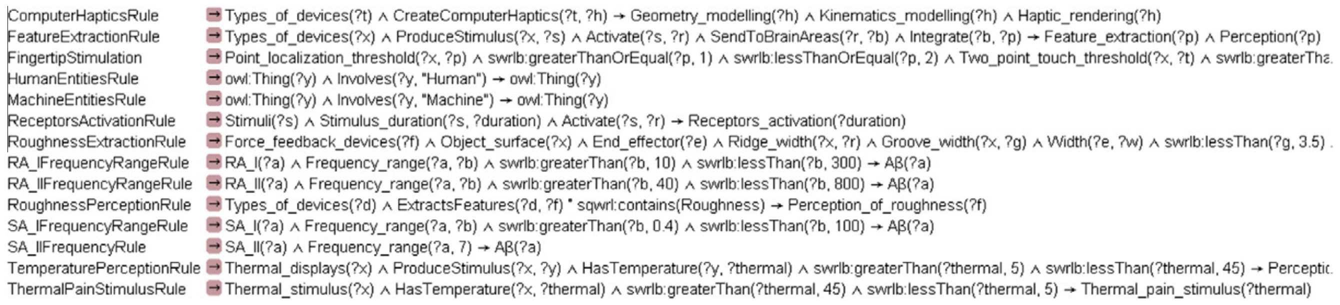
```
Rule 1:  Thermal_displays(?x) ^ ProduceStimulus(?x, ?y) ^ HasTemperature(?y, ?thermal)
         ^ swrlb:greaterThan(?thermal, 5)  ^  swrlb:lessThan(?thermal, 45) →
         Perception_of_temperature-thermal_quality(?x)

Rule 2:  Types_of_devices(?t) ^ CreateComputerHaptics(?t, ?h) →
          Geometry_modelling (?h) ^ Kinematics_modelling(?h) ^ Haptic_rendering(?h)

Rule 3:  RA_I(?a) ^ Frequency_range(?a, ?b) ^ swrlb:greaterThan(?b, 10) ^ swrlb:lessThan(b,300) →
         Aβ(?a)

Rule 4:  Point_localization_threshold(?x, ?p) ^ swrlb:greaterThanOrEqual(?p, 1) ^ swrlb:lessThanOrEqual(?p, 2) ^
         Two_point_touch_threshold(?x, ?t) ^ swrlb:greaterThanOrEqual(?t, 2) ^ swrlb:lessThanOrEqual(?t, 4) →
         Fingertip(?x)
```

Fig. 8. Example rules.



```
ComputerHapticsRule          → Types_of_devices(?t) ∧ CreateComputerHaptics(?t, ?h) → Geometry_modelling(?h) ∧ Kinematics_modelling(?h) ∧ Haptic_rendering(?h)
FeatureExtractionRule        → Types_of_devices(?x) ∧ ProduceStimulus(?x, ?s) ∧ Activate(?s, ?r) ∧ SendToBrainAreas(?r, ?b) ∧ Integrate(?b, ?p) → Feature_extraction(?p) ∧ Perception(?p)
FingertipStimulation         → Point_localization_threshold(?x, ?p) ∧ swrlb:greaterThanOrEqual(?p, 1) ∧ swrlb:lessThanOrEqual(?p, 2) ∧ Two_point_touch_threshold(?x, ?t) ∧ swrlb:greaterTha
HumanEntitiesRule            → owl:Thing(?y) ∧ Involves(?y, "Human") → owl:Thing(?y)
MachineEntitiesRule          → owl:Thing(?y) ∧ Involves(?y, "Machine") → owl:Thing(?y)
ReceptorsActivationRule      → Stimuli(?s) ∧ Stimulus_duration(?s, ?duration) ∧ Activate(?s, ?r) → Receptors_activation(?duration)
RoughnessExtractionRule      → Force_feedback_devices(?f) ∧ Object_surface(?x) ∧ End_effector(?e) ∧ Ridge_width(?x, ?r) ∧ Groove_width(?x, ?g) ∧ Width(?e, ?w) ∧ swrlb:lessThan(?g, 3.5) .
RA_IFrequencyRangeRule       → RA_I(?a) ∧ Frequency_range(?a, ?b) ∧ swrlb:greaterThan(?b, 10) ∧ swrlb:lessThan(?b, 300) → Aβ(?a)
RA_IIFrequencyRangeRule      → RA_II(?a) ∧ Frequency_range(?a, ?b) ∧ swrlb:greaterThan(?b, 40) ∧ swrlb:lessThan(?b, 800) → Aβ(?a)
RoughnessPerceptionRule      → Types_of_devices(?d) ∧ ExtractsFeatures(?d, ?f) ˚ sqwrl:contains(Roughness) → Perception_of_roughness(?f)
SA_IFrequencyRangeRule       → SA_I(?a) ∧ Frequency_range(?a, ?b) ∧ swrlb:greaterThan(?b, 0.4) ∧ swrlb:lessThan(?b, 100) → Aβ(?a)
SA_IIFrequencyRule           → SA_II(?a) ∧ Frequency_range(?a, 7) → Aβ(?a)
TemperaturePerceptionRule    → Thermal_displays(?x) ∧ ProduceStimulus(?x, ?y) ∧ HasTemperature(?y, ?thermal) ∧ swrlb:greaterThan(?thermal, 5) ∧ swrlb:lessThan(?thermal, 45) → Perceptic
ThermalPainStimulusRule      → Thermal_stimulus(?x) ∧ HasTemperature(?x, ?thermal) ∧ swrlb:greaterThan(?thermal, 45) ∧ swrlb:lessThan(?thermal, 5) → Thermal_pain_stimulus(?thermal)
```

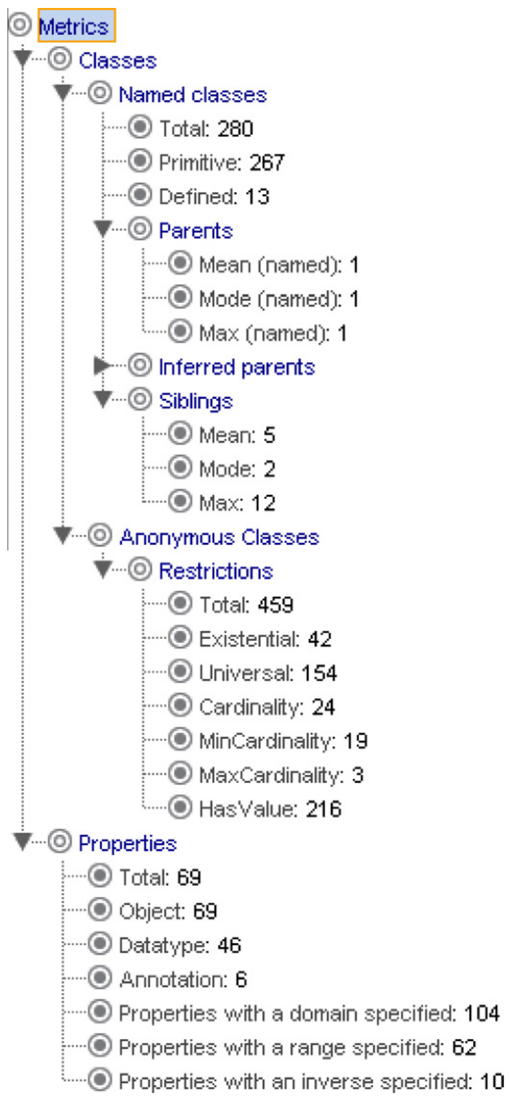
Fig. 9. HASM rules.

receptors has a different frequency range which means that it responds to a range of stimuli frequencies. That data can't be expressed in OWL 1 but can be expressed using SWRL rules. As mentioned before, haptic devices can be designed to be applied not only to hand, palm or fingertips but also to other parts of the body. However, it is important to be aware of the extent to which the cutaneous system is limited by its ability to resolve spatial and temporal details presented to the skin. The spatial resolution is separated to: *two-point touch threshold* (the smallest spatial separation between two stimuli applied to the skin) and *point-localization threshold* (a stimulus is presented to the skin, followed in time by a second stimulus that may or may not be applied to the same site and observers are required to say whether the two stimuli occur at the same or different locations). The values of these properties differ among the skin sites for example: point-localization threshold: (1–2 mm at fingertip and 7–8 mm at palm) and that fact differentiate the haptic devices that are designed for various parts of the body (Lederman & Klatzky, 2009).

The description of classes is completed when the types of properties and the types of restrictions are defined. Fig. 5 depicts the definition of class hasm:Types-of-devices. Bandwidth, crosstalk, degrees of freedom, device frequency, device resolution, force, crosstalk, friction, peak force, precision, motion range and sensation modality are datatype properties that their values define the operation and performance of the device. For example, device-frequency is a datatype property and has a cardinality restriction min 1, that means that a haptic device can produce one or more frequency values. The restriction of property produceStimulus *not* (*ProduceStimulus only Pain-stimulus*), means that every haptic device can produce mechanical, thermal, electrical, force but not pain stimulus. Fig. 6 depicts the definition of class hasm:Force-feedback-devices which is a subclass of hasm:Types-of-devices class. Force-feedback-devices class is defined by the object properties: *produceStimulus*, *extractFeatures* and *hasSensors* and the datatype property *Sensation-modality* that can have the values muscle-stretch, vibration or joint-tension. Fig. 7 depicts the definition of hasm:Mechanoreceptors class that is a subclass of Receptors class and presents the necessary and inherited conditions. Instances of mechanoreceptors class do not have a *connected-Through* relation to instances that are not members of the class Aβ (connectedThrough only Aβ) and have at least one *activatedBy* relation with instances of the classes of mechanical or electrical stimulus (activatedBy some (Mechanical-stimulus or Electrical-stimulus)). Density, frequency range and sensation modality are datatype properties that define receptors. The cardinality restrictions depict that density has more than one value because it depends on the part of the body that the receptors are located (the larger number of mechanoreceptors are in the fingertip). Moreover, each class of receptors responds to a different range of frequencies (Frequency-range min 1).

5.3. SWRL rules

Rule technology has been around for decades, has found extensive use in practice, and has reached significant maturity. Rules provide the capability to express more complex relationships and restrictions between concepts and have more precise control of the reasoning process (Antoniou & van Harmelen, 2008). Specifically, rules model the dynamic aspects of complex relationships and restrictions of classes. For instance, each class of skin receptors responds to a specific range of frequencies while haptic devices generate stimuli that correspond to different frequency ranges. The complexity of such restrictions cannot be expressed sufficiently with OWL alone. Due to this fact, the HASM OWL ontology needed to be extended with SWRL rules. As analyzed in a previous section, ontologies are based on Description Logics, while rules are based on logic programming. Given that interoperability is one of the primary goals of the Semantic Web and that rules are a key part of these goals, there has been significant recent interest in standardization (Connor et al., 2005). The goal of sharing and



```
Metrics
  Classes
    Named classes
      Total: 280
      Primitive: 267
      Defined: 13
      Parents
        Mean (named): 1
        Mode (named): 1
        Max (named): 1
      Inferred parents
      Siblings
        Mean: 5
        Mode: 2
        Max: 12
    Anonymous Classes
      Restrictions
        Total: 459
        Existential: 42
        Universal: 154
        Cardinality: 24
        MinCardinality: 19
        MaxCardinality: 3
        HasValue: 216
  Properties
    Total: 69
    Object: 69
    Datatype: 46
    Annotation: 6
    Properties with a domain specified: 104
    Properties with a range specified: 62
    Properties with an inverse specified: 10
```

Fig. 10. HASM metrics.

exchanging rule bases while still being able to process them using different rule engines has resulted in RuleML,[6] SWRL,[7] Metalog,[8] ISO Prolog[9] and other standardization efforts.

In order to generate rules for the HASM we used SWRL (Semantic Web Rule Language), that allows users to write Horn-like rules expressed in terms of OWL concepts to reason about OWL instances. The rules can be used to infer new knowledge from existing OWL knowledge bases. Rules in SWRL reason about OWL instances in terms of OWL classes and properties. Rules cannot define classes and properties of an ontology neither create new objects but rather they can derive values of properties (object and datatype) for existing instances or they can re-classify existing instances to more specific classes, based on complex applications specific semantics. HASM rules were generated with Protege SWRL Editor that is a plugin in Protege environment and with the support of the Jess Rule Engine[10].

In HASM there have been developed classification rules and rules that define datatype properties values. Fig. 8 presents two classification rules and two derived attributes rules. Rule 1 describes the perception of thermal flow through thermal tactile displays. A thermal display produce a stimulus (produceStimulus relation) and a stimulus has temperature (hasTemperature datatype property). The temperature stimulus must have a value of the hasTemperature property between 5 and 45 °C. Rule 2 defines that for every device, computer haptics must include haptic rendering, geometry modeling and kinematics modeling processes. Rule 3 defines the frequency range that activates the RA-I receptors. RA-I receptors respond to a stimulus that its frequency is between 10 and 300 Hz. Rule 4 defines the data for the point localization and two point threshold attributes in the fingertip. Specifically it defines the ability of the fingertip to perceive a single stimulus or two stimulus applied separately. Fig. 9 presents a snapshot of the SWRL editor and presents the set of SWRL rules for HASM.

5.4. Ontology metrics

Finally, Fig. 10 presents the ontology metrics for the HASM ontology that include number of classes, properties and restrictions. Moreover, the number of HASM instances was 199, the number of SWRL rules was 15, the number of OWL axioms exported to the rule engine was 915, the number of imported axioms was 430 and the number of inferred axioms was 83.

6. Case study: roughness representation

Among the various perceptual properties that characterize object surfaces, roughness has undoubtedly received the most attention from haptic researchers. The roughness percept reflects the properties of the surface touched in interaction with the manner in which the surface/object is explored (Lederman & Klatzky, 2009). Based on the standards for roughness perception through a rigid link we designed using CHAI 3D, a rough surface. Consequently using a specific haptic device (Novint Falcon) we attempt using HASM and the built-in SWRL roughness rule to find the pathway of haptic interaction from haptic device to the extraction of the object's characteristic. This case study provides a proof of concept of how flexible and useful HASM is connecting the stimulus to the perception and thus providing a practical tool in the haptic software design process. In this section we describe a case study for HASM involving instances and SWRL rules, relative to the perception of roughness generated by Novint Falcon[11] which is a ground based haptic device. Novint Falcon is a force feedback haptic device and allows the user to manipulate virtual objects through a rigid probe. The device translates the user's motions and sends to the user feedback according to the surface of the objects that he/she explores. The device simulates haptic properties such as texture, roughness, shape, edges and compliance. Roughness characterizes the texture of a surface using the roughness measure and is quantified by the vertical deviation of a real surface from its ideal form (smooth). If these deviations are large, the surface is rough; if they are small the surface is smooth. The texture of a surface and especially its roughness can be perceived through devices that produce force feedback or tactile feedback. The human perception of roughness can concern real surfaces with roughness that a user can touch with bare hand, real surfaces with roughness that a user can explore through stylus or probe and virtual textures that are generated from haptic devices with force feedback and tactile displays (Lawrence, Kitada, Klatzky, & Lederman, 2007). The perception of roughness is realized with stimuli based on pressure and vibration.

[6] http://www.ruleml.org/.
[7] http://www.w3.org/Submission/SWRL/.
[8] http://www.w3.org/RDF/Metalog/.
[9] http://www.univ-orleans.fr/lifo/software/stdprolog/docs.html/.
[10] http://www.jessrules.com/.
[11] http://www.novint.com/index.php/home.

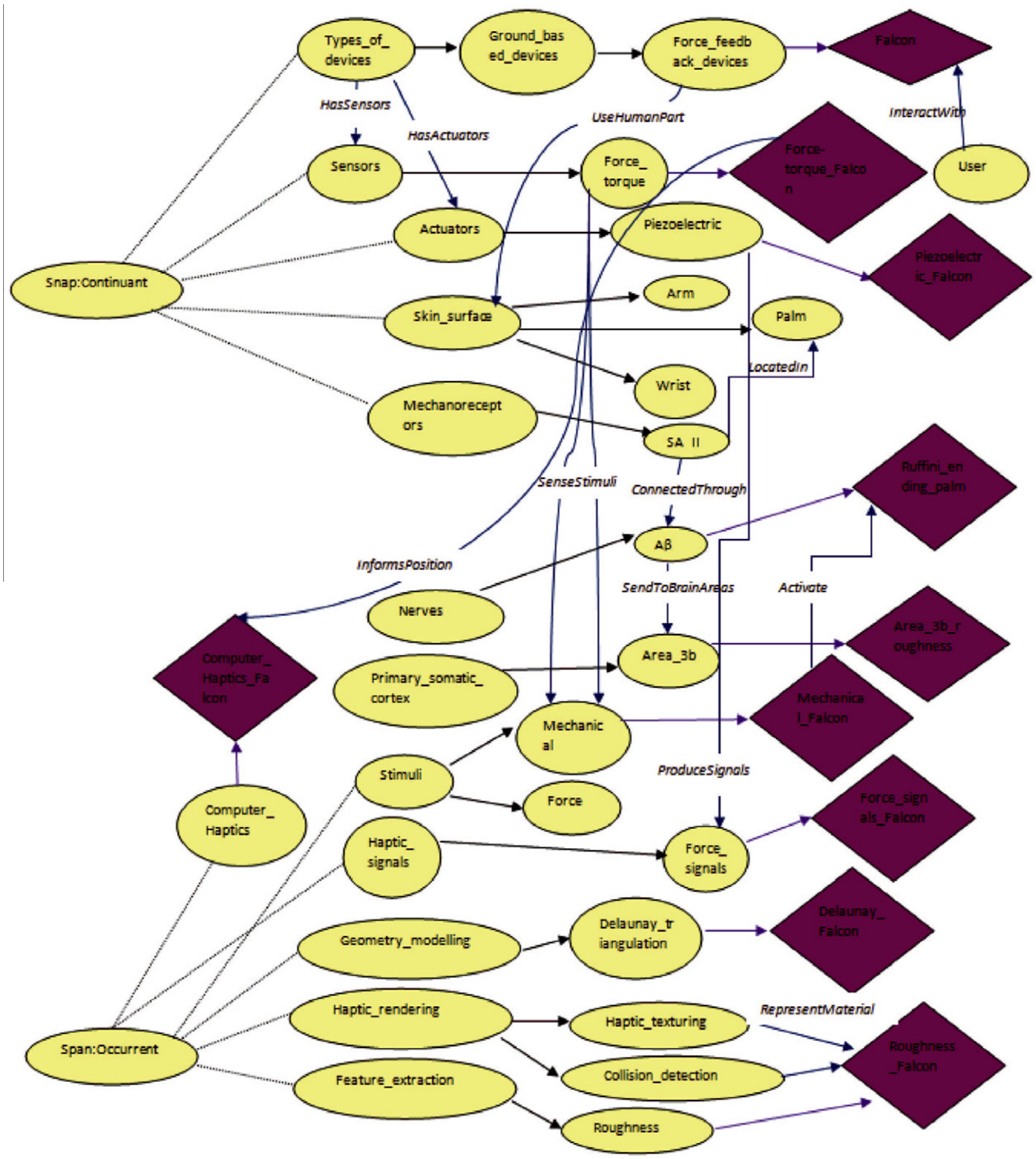


Fig. 11. Falcon instance.

6.1. Instances

OWL classes are interpreted as sets of instances. By creating instances of HASM classes and given a haptic device we can determine which classes are needed to describe the interaction with the given device and the pathways of haptic information flow in order to simulate the perception of a haptic property. Fig. 11 presents the haptic interaction with the Novint Falcon device and the perception of roughness with this device. Novint Falcon is an instance of the class Force-feedback devices. For the case study, 39 instances have been created that belong to 25 classes. Especially the instances that have been created and the relations that have been used for the case study are:

- Novint-Falcon *produceStimulus* Mechanical-stimulus-falcon, Force-stimulus-falcon
- Mechanical-stimulus-falcon *activate* Merkel-cell-palm, Ruffini-ending-palm
- Merkel-cell-palm, Ruffini-ending-palm *connectedThrough* Aβ
- Aβ *locatedIn* Palm-falcon
- Merkel-cell-palm, Ruffini-ending-palm *sendToBrainAreas* Area-3b-roughness-falcon

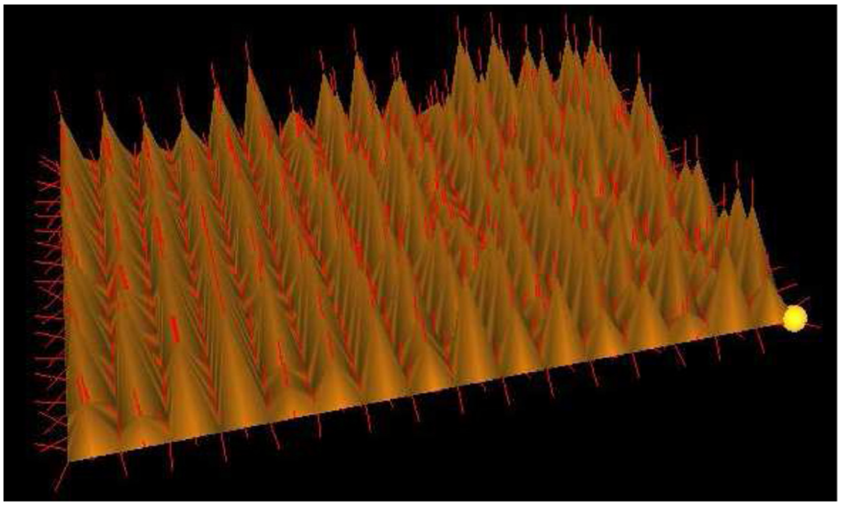
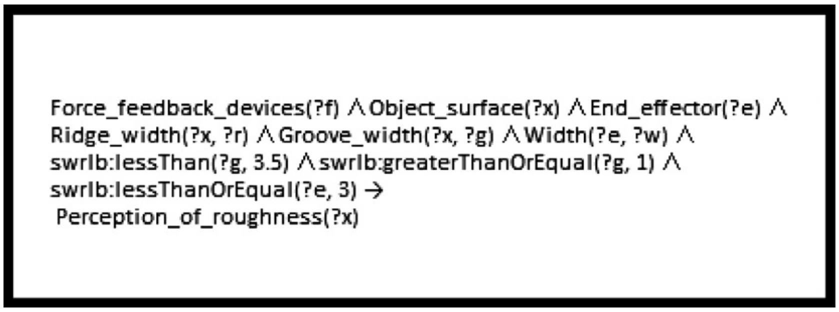


Fig. 12. Surface with roughness using mesh generation with Novint Falcon.



```
Force_feedback_devices(?f) ∧ Object_surface(?x) ∧ End_effector(?e) ∧
Ridge_width(?x, ?r) ∧ Groove_width(?x, ?g) ∧ Width(?e, ?w) ∧
swrlb:lessThan(?g, 3.5) ∧ swrlb:greaterThanOrEqual(?g, 1) ∧
swrlb:lessThanOrEqual(?e, 3) →
 Perception_of_roughness(?x)
```

Fig. 13. Roughness perception rule.

- Piezoelectric-actuator-phantom *produceSignals* Force-signals-falcon
- Novint-Falcon *hasSensors* Force-torque-falcon
- Novint-Falcon *produceFeedback* Force-feedback-falcon
- Force-torque-falcon *informsPosition* Computer-Haptics-Falcon
- Axis-Aligned-Bounding-Boxes-falcon *representMaterial* Roughness-falcon
- Haptic-texturing-falcon *representMaterial* Roughness-falcon
- Plastic-virtual-objects-falcon *representMaterial* Roughness-falcon

6.2. Roughness model with CHAI 3D

Using Novint Falcon device and the CHAI 3D[12] which is an open source set of C++ libraries for computer haptics, visualization and interactive real-time simulation, we have built a surface with roughness. CHAI 3D provides the methods for haptic software modeling such as: haptic rendering, kinematics, geometry modeling and collision detection. The application that we have built indicates the interaction between the device and a surface with roughness and simulates the perception of roughness. The model of the surface that has been generated for the application is presented in Fig. 12.

A surface with roughness is a surface that includes grooves and ridges. Among the important behavioral findings is that surface roughness is primarily determined by the inner spacing between the elements that constitute the surface and is affected by the force of exploration. Perceived roughness magnitude increases monotonically with increasing spacing (groove width) until it reaches approximately 3.5 mm. The width of the ridges that constitute the surface has small perceptual effect (Klatzky & Lederman, 2008). These findings are also confirmed from our simulation. The perception of roughness through a haptic device is less precise than a tactile display which uses the fingertips that have the finest spatial and temporal resolution for texture perception.

Roughness findings differ for manual exploration of a surface (exploration with bare finger) and exploration through a rigid link or a probe (Novint Falcon). The interelement spacing where perceived roughness reaches a maximum is related to the width of the exploring end effector (approximately 3 mm). Based on the findings for roughness perception through a rigid link, and the results of our simulation with Novint Falcon, a SWRL rule is built that describes how a surface with roughness can be designed. This rule is presented in Fig. 13 and can be used as an input to the HASM ontology in order to provide the pathway of haptic interaction from a haptic device to the extraction of the object's characteristic. The roughness rule sets the limits of groove width, ridge width and end effector (rigid link) width of a surface in order to perceive roughness through a haptic device.

7. Conclusions

In this paper we have described the development of HASM in OWL, an ontology for haptic applications software modeling, in relation to the entities that take part in the tactile information flow

[12] http://www.chai3d.org.

during human-haptic interaction, as well as a set of SWRL rules that complement the ontology in representing the dynamic inference aspects of the information flow in the ontological representation of such a hybrid interconnected system. The ontology design is based on the two systems that compose the haptic interaction: the human system and the machine system. The entities that each system includes are analyzed and they are classified into object classes and process classes.

HASM structure is a specialization of the Basic Formal Ontology (BFO), which is an upper-level ontology that can be used in support of domain ontologies. Ontology development has been done with the Protege ontology editor in the Ontology Web Language (OWL) which provides to the growing software user community a suite of tools to construct domain models and knowledge-based applications, and enables the exportation of ontologies in many different formats.

The proposed ontology was developed to address some open issues. First, we formalized the vocabulary that describes human-haptic system interaction, providing a formal classification of the haptics domain that can be utilized by users and applications. Second, the ontology we have developed for the haptics will hopefully help in designing better software for tactile interfaces. Moreover, we believe that the proposed ontology could serve as a framework to allow an easy acquisition of knowledge about haptics domain and human-haptic interaction.

Except from the hierarchical taxonomy of haptic entities into classes, the ontology includes the relations between classes that are analyzed and described in detail. The definition of relations between entities is a complete description of the haptics domain, assisting the understanding of the different kind of haptic interactions and is the basis for the design of instances and SWRL rules. We also analyzed the rules that have been added to the ontology in order to extend its reasoning capacity. The set of rules along with the OWL properties can be used from users in order to design a complete haptic interaction model that satisfies the requirements for a specific haptic interface or device.

Finally, we have presented a case study of a set of instances that concern haptic interaction with a specific haptic device. The device that has been used is the Novint Falcon device that produces force feedback to the user. Moreover, given a specific haptic device such as Novint Falcon and a haptic characteristic of an object such as roughness, we used the instances that come from the HASM classes, the rules for force feedback devices and roughness perception, along with the CHAI 3D haptic library in order to built an application that simulates a haptic interaction with a force feedback haptic device and a virtual object with roughness. The use of HASM in software design for an application demonstrated the pathways that haptic information follows between human and machine system in order to simulate efficiently the perception of an object.

Future research is directed to the design and development of a knowledge-based system or a semantic web portal that will exploit the ontological information of HASM through semantic searching and reasoning and will be used for consulting haptic software/system engineers, in order to build efficient haptic applications. This portal can also be used as a system that the haptic community will use to add new devices, new knowledge about novel technologies or modeling methods in the haptics domain.

Acknowledgements

This paper is part of the 03ED375 research project, implemented within the framework of the "Reinforcement Programme of Human Research Manpower" (PENED) and co-financed by National and Community Funds (25% from the Greek Ministry of Development - General Secretariat of Research and Technology and 75% from E.U. - European Social Fund).

References

Alamri, A., Eid, M., & Saddik, A. (2006). Towards a standard modeling of haptic software system. In *IEEE international workshop on haptic audio visual environments and their applications* (pp. 84–88).

Alamri, A., Eid, M., & Saddik, A. (2007). A haptic enabled uml case tool. In *IEEE international conference on multimedia and expo*.

Antoniou, G., & van Harmelen, F. (2008). *A semantic web primer*. MIT Press.

Barbagli, F., & Salisbury, K. (2006). Haptic discrimination of force direction and the influence of visual information. *ACM Transactions on Applied Perception, 3*, 125–135.

Basdogan, C., & Srinivasan, M. A. (2002). *Handbook of virtual environments: Design, implementation and applications*. Lawrence Erlbaum Associates. Chapter Haptic rendering in virtual environments, pp. 117–134.

Biggs, S. J., & Srinivasan, M. A (2002). *Handbook of virtual environments: Design, implementation and applications*. Lawrence Erlbaum Associates. Chapter Haptic interfaces, pp. 93–115.

Bittner, T., & Barry, S. (2004). Normalizing medical ontologies using basic formal ontology. In *Proceedings of GMDS* (pp. 199–201).

Bonacin, R., Baranauskas, C. C., & Liu, K. (2004). From ontology charts to class diagrams: Semantic analysis aiding systems design. In *Proceedings of the sixth international conference on enterprise information systems* (pp. 389–395).

Brauer, M., & Lochmann, H. (2008). In *ESWC 2008*. Chapter An ontology for software models and its practical implications for semantic web reasoning (Vol. 5021, pp. 34–48), Springer, Heidelberg.

Chandrasekaran, B., Josephson, J. R., & Benjamins, R. V. (1999). What are ontologies and why do we need them? *IEEE Intelligent Systems, 14*, 20–26.

Chouvardas, V., Miliou, A., & Hatalis, M. (2008). Tactile displays: Overview and recent advances. *Displays, 29*, 185–194.

Connor, M. J., Knublauch, H., Tu, S. W., Grossof, B., Dean, M., Grosso, W. E., & Musen, M. A. (2005). Supporting rule system interoperability on the semantic web with swrl. In *Fourth international semantic web conference*.

Coutaz, J., Lachenal, C., & Dupuy-Chess, S. (2003). Ontology for multi-surface interaction. *Human–Computer Interaction*.

Cranefield, S., & Purvis, M. (1999). Uml as an ontology modeling language. In *Proceedings of the workshop on intelligent information integration, 16th international joint conference on AI (IJCAI-99)* (pp. 46–53).

Devedzic, V. (2002). Understanding ontological engineering. *Communications of the ACM, 45*, 136–144.

Eid, M., Alamri, A., Melhem, J., & Sa, (2008). Evaluation of uml case tool with haptics. In *HAS 08*.

Evermann, J., & Wand, Y. (2005). Ontology based object-oriented domain modelling: Fundamental concepts. *Requirements Engineering, 10*, 146–160.

Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., & Schneider, L. (2002). Sweetening ontologies with dolce. In *Proceedings of EKAW* (pp. 166–181).

Gardner, E., & Kandel, E. (1991). *Principles of neural science* (4th ed.). Elsevier. Chapter Touch, pp. 452–472.

Gardner, E., Martin, J., & Jessel, T. (1991). *Principles of neural science* (2nd ed.). Springer. Chapter The bodily senses, pp. 431–451.

Grenon, P. (2003). Spatio-temporality in basic formal ontology, part I: SNAP and SPAN, upper-level ontology and framework for formalization. Technical Report. IFOMIS group.

Grenon, P., Smith, B., & Goldberg, L. (2004). *Ontologies in medicine*. IOS Press. Chapter Biodynamic ontology: Applying BFO in the biomedical domain, pp. 20–38.

Hale, K. S., & Stanney, K. M. (2004). Deriving haptic design guidelines from human physiological, psychophysical and neurological foundations. *IEEE Computer Graphics and Applications, 24*, 33–39.

Hayward, V., & Astley, O. R. (1996). Performance measures for haptic interfaces. In *Robotics research: The 7th international symposium* (pp. 195–207).

Ho, C., Basdogan, C., & Srinivasan, M. A. (1999). Efficient point-based rendering techniques for haptic display of virtual objects. *Presence*.

Hruby, P. (2005). Role of domain ontologies in software factories. In OOPSLA 2005 workshop on software factories.

Iggo, A. (1984). Cutaneous receptors and their sensory functions. *The journal of hand surgery, 9*, 7–10.

Kinashi, S., Sugisaki, Y., Kanao, H., Fujisawa, M., & Miura, K. (2005). Development of a geometric modeling device with haptic rendering. In International CAD Conference and exhibition.

Klatzky, R., & Lederman, S. (2008). *Haptic rendering: Foundations. Algorithms and applications*. A K Peters, Ltd.. Chapter Perceiving object properties through a rigid link, pp. 7–19.

Lawrence, M. A., Kitada, R., Klatzky, R. L., & Lederman, S. J. (2007). Haptic roughness perception of linear gratings via bare finger or rigid probe. *Perception, 36*, 547–557.

Laycock, S. D., & Day, A. M. (2003). Recent developments and applications of haptic devices. *Computer Graphics, 22*, 117–132.

Lederman, S. J., & Klatzky, R. L. (2009). Haptic perception: A tutorial. *Attention, Perception and Psychophysics, 71*, 1439–1459.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., & Schneider, L. (2003). The wonderweb library of foundational ontologies. Technical Report. ISTC-CNR.

Mokos, K., Meditskos, G., Katsaros, P., Bassiliades, N., & Vasiliades, V. (2010). Ontology-based model driven engineering for safety verification. In *36th EUROMICRO conference on software engineering and advanced applications (SEAA 2010)* (pp. 47–54). IEEE Computer Society.

Myrgioti, E., Chouvardas, V., Miliou, A., & Hatalis, M. (2007). Modeling tactile information flow using ontologies. In *Third balkan conference in informatics (BCI 07)* (pp. 219–228).

Myrgioti, E., Miliou, A., & Chouvardas, V. (2009). Ontological representation of tactile information for software development. *Applied Ontology, 4*, 139–167.

Niles, I., & Pease, A. (2001). Towards a standard upper ontology. In C. Welty, B. Smith (Eds.), *Second international conference on formal ontology in information systems (FOIS-2001)* (pp. 2–9).

Noy, N., Sintek, M., Decker, S., Crubezy, M., Fergerson, R., & Musen, M. (2001). Creating semantic web contents with protege-2000. *Intelligent Systems, IEEE, 16*, 60–71.

Noy, N. F., & McGuiness, D. (2001). Ontology development 101: A guide to creating your first ontology. *Knowledge Systems Laboratory*.

Obrenovic, Z., & Starcevic, D. (2004). Modeling multimodal human–computer interaction. *IEEE Computer, 37*, 65–72.

Obrenovic, Z., Starcevic, D., & Devedzic, V. (2003). Using ontologies in design of multimodal user interfaces. In M. Rauterberg (Ed.), *Human–computer interaction – INTERACT'03* (pp. 535–542). IOS Press.

Ruiz, F., & Hilera, J. (1998). Springer. Chapter Using ontologies in software engineering.

Salisbury, K., Conti, F., & Barbagli, F. (2004). Haptic rendering: Introductory concepts. *IEEE Computer Graphics and Applications, 24*, 24–32.

Shegogue, D., & Zheng, W. J. (2005). Integration of the gene ontology into an object-oriented architecture. *BMC Bioinformatics, 6*, 113.

Spear, A. D. (2006). Ontology for the twenty first century: An introduction with recommendations. Internal Report. Institute for formal ontology and medical information science (IFOMIS), Saarbrucken, Germany University at Buffalo, Buffalo, NY, USA.

Srinivasan, M., Cutkosky, M., Howe, R., & Salisbury, J. (1999). Human and Machine Haptics. Technical Report. MIT Press, Cambridge, Mass.

Srinivasan, M. A., & Basdogan, C. (1997). Haptics in virtual environments: Taxonomy, research status, and challenges. *Computers & Graphics, 21*, 393–404.

Wongthongtham, P., Chang, E., Dillon, T. S., & Sommerville, I. (2005). Software engineering ontologies and their implementation. In P. Kobol, (Ed.), *IASTED International conference on software engineering (SE)* (pp. 208–213).