DIAGNOSIS OF AMBIENT SYSTEMS BASED ON THE MODELING OF EFFECTS

Ahmed Mohamed [1]) Christophe Jacquet [1]) Yacine Bellik [2])

Abstract

Complex ambient systems are composed of many heterogeneous systems; each system interacts with its surroundings using collected environmental data from sensors' readings. Diagnosis for the overall system as well as the sub-systems composing it must address many challenges caused by the dynamic nature of these systems and the impossibility to pre-define control loops between sensors and actuators at design time as they are discovered dynamically at run-time. This paper presents a design technique that is based on defining for each component the expected effect — the "physical phenomenon" — it is supposed to produce or receive. The proposed approach facilitates the comparison of the produced effects in the environment and the actual readings of the sensors, and thus simplifies the diagnosis task. To do so, we propose a precise definition of the concept of effect. The approach is validated by implementing a simple yet complete example taking place in an ambient environment.

1. Introduction

Ubiquitous systems are a particular type of interactive systems in which information processing and communication capacities have been integrated into everyday objects and activities (ubiquity). Users of ubiquitous computing are not necessarily aware that they are so. These systems are usually composed of many heterogeneous complex systems. Unlike workstations whose actions only affect themselves and their immediate surroundings, ambient systems incorporate devices that enable them to act more broadly on their physical environment. These devices are called actuators. Moreover, ambient systems are aware of their environment by collecting local data using sensors. After processing these data, they may change the conditions of their environment using actuators in order to satisfy user's preferences or to assist him/her in his/her task (ambient intelligence). In this context, the system must have the means to check autonomously whether the system actions are

[1] SUPELEC Systems Sciences (E3S) – Department of Computer Science – 3 rue Joliot-Curie 91192 Gif-sur-Yvette Cedex, France, {ahmed.mohamed | christophe.jacquet}@supelec.fr

[2] LIMSI-CNRS – BP 133 91403 Orsay Cedex, France, yacine.bellik@limsi.fr

* This work has been performed within the CBDP project, a project co-funded by the European Union. Europe is involved in Région Île-de-France with the European Regional Development Fund

done correctly. In software, when calling a procedure, mechanisms such as exceptions and error codes allow the caller to decide whether the execution was successful or not. As a matter of fact, when the ambient system sends out orders to an actuator, the latter can provide a return code, but the information provided by this code reflects only the way the orders are transmitted to the actuator, not their actual execution. For instance when the system activates a light bulb, it can receive an acknowledgment enabling it to know that the order was successfully transmitted to the electrical circuit. However it doesn't know if the light has really been switched on (for instance due to a damage to the bulb itself). Even though control theory applied to embedded systems allows one to pre-determine closed control loops using ad-hoc sensors, the particularity of ambient systems is that physical resources, mainly sensors and actuators, are not necessarily known at design time. In fact they are dynamically discovered at run-time, so control loops cannot be pre-determined. Therefore the main goal of our work is to provide a dynamic and reliable method for building dynamically the equivalent of control loops for such systems, using available sensors at a given time, in order to perform an accurate diagnosis at run-time. We propose an approach in which the characteristics of actuators and sensors, as well as the effects produced by the actuators and captured by the sensors, are precisely described. The concept of effect is central to this approach: using this concept, an ambient system is capable of automatically associating actuators and sensors, and thus, of deducing the expected measurement provided by a given sensor when a certain action is performed by an actuator. The remainder of this paper is organized as follows. Section 2 reviews prior studies for performing diagnosis in ambient environments. Section 3 introduces the concept of *effect*. Section 4 explains how this definition is used to model and build an ambient system. Section 5 presents a complete example demonstrating our approach.

2. Related Work

One of the major challenges studied in previous work is the fact that ubiquitous systems are supposed to operate completely in the background in such a way that the users are not supposed to notice their existence. This very important characteristic of ubiquitous computing makes fault detection and diagnosis a difficult task [1]. Besides, with such requirements, the behavior of an ambient environment creates a risk which is that the user may continue to rely on failed services without noticing 0. Some ubiquitous computing environments, especially smart homes, also called assisted living homes, use an approach that consists in making the diagnosis task user-centered. As mentioned earlier, the main purpose of ubiquitous environments is to discreetly satisfy the user's preferences. Based on this principle most of the proposed diagnosis techniques start with gathering data which are processed to generate a text-based summary [3], used for the diagnosis. The generated summary is then made available to human intervention (the user, an expert, a service technician, etc.). We note that the user is usually at the center of the diagnosis as the latter depends on the user's feedback. Another approach proposed for ambient environment diagnosis is the model-

based diagnosis technique which is based on first-principles reasoning by means of a system description that is used to define the behavior of each component within the system and the connectivity of these components [4]. With this information, a diagnosis can be made by simulating the system's behavior. In fact the major challenge of this technique is combinatorial explosion which makes the approach useless for devices composed of a considerable number of components [5]. We can say that generally the approaches proposed in existing work suppose that sensors and actuators are somehow directly linked. In other words the model explicitly contains the relationships that link actuator actions and sensor states. We claim that building such explicit links is poorly adapted to highly dynamic ambient systems. Indeed, as devices are added to and removed from an environment at runtime, it is very difficult for the system designer to thoroughly describe the system at design time. For these reasons, we introduce the concept of "effect" that will allow actuators and sensors to be decoupled in the model, while enabling the system to deduce the links between them at runtime.

3. The Concept of "Effect"

The purpose of the concept of effect is to enable the simulation of the physical consequences of an action in an ambient environment. In the proposed approach the effect becomes the only link between the actuators and the sensors. As a matter of fact, in an ambient environment, actuators, when receiving orders sent by the system, emit actions whose actual effects on the surrounding environment are only visible to the system through its sensors. We consider these actions from the point of view of the physical environment; accordingly they are defined as one or more physical phenomenon. The format, the value and the way these actions are perceived by the sensors and processed by the system follow the corresponding physical laws. These laws depend on a number of well-known physical parameters. In the proposed approach an explicit list of the effects that are expected to be observed in the environment must be defined and modeled by the system designer. For each declared effect a number of properties are defined too. These properties, which we will call effect properties from now on, correspond to the physical parameters defining the physical laws mentioned before. If we observe carefully the nature of the data collected by sensors, we conclude that these effect properties match exactly what sensors are sensible to. Even though the list of properties can be very explicit, it is to be noted that in reality the number and the nature of the properties are conditioned by the hardware configuration (the nature of the sensors and their precision) and the degree of details wanted. Our motivation for proposing such a definition is to be able to decide whether or not an actuator has completed an action successfully, by applying the physical rules on the effects produced by actuators on one hand, and on the actual effect properties detected by the sensors on the other hand. An important point to note here is that the definition of the effect can follow different levels of granularity. For instance the light emitted from a light bulb in an ambient environment might be modeled using classical laws of physics for light propagation

[6], or a very simple rule saying “if a light bulb is on in a room then the light sensors that are in that room should detect light”. Likewise an actuator that produces more than one effect can have its generated effects defined according to different levels of details. The choice of the latter can depend, among other things, on the context of use, for instance assisted living homes for blind persons would have a very detailed definition of the model for the propagation of sound waves. Moreover, several different devices may contribute to one given effect. Therefore, this effect may be defined more than once according to different level of details, depending on the importance of the produced effect with respect to each device. For example the heating effect generated by light bulbs is not as significant as the heating effect generated by heaters; nevertheless it might be important to model the light bulb heating effect in a particular scenario. We can push the model further by defining the same effect produced by the same device more than once with different levels of details. The diagnosis results from the different levels can be useful for the system’s overall diagnosis. This generality and flexibility of the definition allows us to have more or less realistic definitions of the physical laws depending on different criteria such as the architecture of the system, the diagnosis technique used for the system, how accurate we want the diagnosis to be, the desired level of detail we want for the diagnosis, the context of use of the ambient environment, etc. Such flexibility is well-suited to the nature of ambient environments. The definition of the effects allows us to apply calculations based on pre-defined physical rules in order to calculate the expected readings of the sensors. Once the expected results have been determined, the final step would be to perform a diagnosis by comparing the actual readings of the sensors with the calculated expected readings. It should be noted that it is up to the final designer of the system to define the appropriate physical rules and thus decide about the desired level of details of each physical rule applicable in the environment. What we propose here is a generic model that is adaptable to different levels of granularity. It is also important to note that the physical law defines not only the calculation formula but also the way that the physical phenomena interfere with each other.

4. A Meta-Model for Diagnosis

4.1. Supporting Ontology

Section 3 has shown that effects are a solution to avoid any predefined actuator-sensor relations; instead an automatic deduction of these relations is made possible. The latter deduction is based on physical laws that calculate the expected values for the sensors in order to perform a diagnosis process. The manner in which the diagnosis process should be managed must be flexible as it is supposed to support centralized as well as distributed diagnosis process, in which case every device would have a local diagnosis process. The latter may be a partial diagnosis solution for large systems, using local observations [7] as well as an overall diagnosis algorithm. To benefit from good extensibility properties and broad tool support, we have used ontologies for defining the meta-

model and the models, namely OWL ontologies [8]. The ontology schema is given in Figure 1. For clarity reasons the main branches of the ontology are detailed separately in the next paragraphs.

4.1.1. The Ontology's Main Branches

- Concept of "effect"

One of the main objectives of our approach, illustrated by Figure 2, is to eliminate the direct link between sensors and actuators. As a matter of fact the possible relationship between an actuator and a sensor is indirectly specified through the use of an *effect*. An effect is characterized by a certain number of properties. The definition of the properties is conditioned, on the one hand by the nature of the effect, and on the other hand by the desired degree of details. For instance, in a coarse-grained model the effect of human voice may be characterized by the sound level only, whereas in a finer-grained model it may be characterized by its frequency distribution as well. By definition, the purpose of an actuator is to produce one or more effects. Likewise, the purpose of a sensor is to detect a property of the effect.

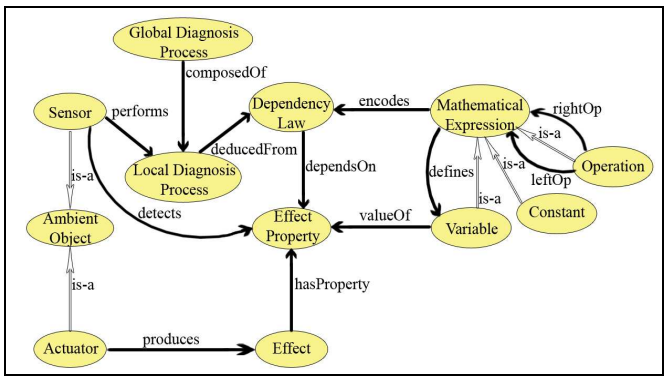

Figure 1. The ontology schema

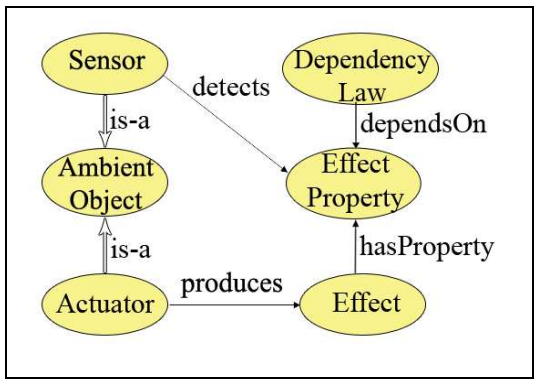

Figure 2. Definition of effects in the ontology

- Support for the diagnosis process

We have made the choice to associate to each sensor a diagnosis process node, as shown in Figure 3. This node is to contain the result of the local diagnosis which is based, in part, on the local readings of the corresponding sensor. As explained earlier, this choice brings certain flexibility to the diagnosis architecture of the whole system as it leaves the way open for using any kind of diagnosis strategy, especially for networked systems. In addition, each diagnosis process deduces its diagnosis results also from one or many dependency laws, which represent the "physical" laws applicable around the sensor and estimate the expected values for the *effect properties* read by the sensor. The relation between diagnosis process and dependency law is also illustrated in Figure 3.

- The Dependency law

Dependency laws, on which diagnosis processes rely, depend in their turn on mathematical formulae. The latter, along with the variables and the operators, based on simple object oriented inheritance principles, constitute the computational model [9], as opposed to the physical model represented by the dependency laws; Figure 4. The computational model, as its name indicates, helps calculating the estimated values of the effect properties that sensors are supposed to read by applying the corresponding dependency laws. As mentioned earlier, the computational model may also have different possible levels of details, depending on the types of actuators and sensors used in a given system, and on the objectives of the system designer.

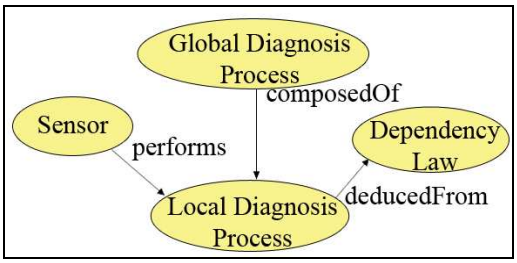

Figure 3. The diagnosis process

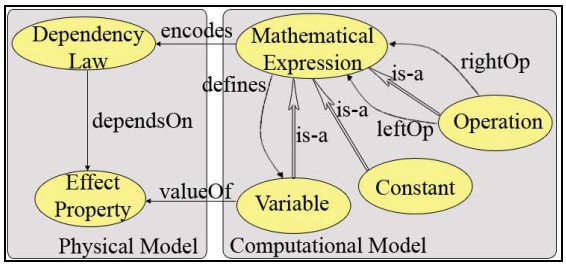

Figure 4. Physical model vs computational model

4.1.2. The reasoning engine

So far we have presented a way to model actuators and sensors, as well as a mechanism to deduce links between them (effects), but what remains to be seen is the way in which these effective links may be deduced, allowing the system to eventually perform a diagnosis. We have chosen to write deduction rules in first-order logic, as it is well-supported by OWL off-the-shelf tools such as Jena [10]. The generic rule below figures out which is the actuator that generates an effect whose parameters are detected by a sensor. Such a rule may attach an error message to the "DiagnosisProcess" node. In a syntax inspired by that of Jena, the rule would be:

(?Sensor *detects* ?EffectProperty) ^ (?Effect *hasProperty* ?EffectProperty) ^ (?Actuator *produces* ?Effect) ^ (?Sensor *performs* ?DiagProcess) ^ (?DiagProcess *deducesFrom* ?DependencyLaw) ^ (?DependencyLaw *dependsOn* ?EffectProperty) ^ (?DiagProcess *value* "False") → (?DiagProcess *message* "Please check the "+?Actuator)

This means that: if a sensor detects an effect property, if an effect that is produced by an actuator has the same effect property, if the diagnosis process that is performed by that sensor and deduced from a dependency law depending on the effect property in question fails, then we generate an error message suggesting to check the actuator that generated the effect.

4.2. Proposed Modeling Methodology

In this paper we propose general guidelines for the designer of an ambient system to set up diagnosis for this system. A possible output for the proposed technique would be an ontology-driven application [11]. Thus the related ontology would be considered as the application's architectural artifact [11]. In other words, the ontology would be considered as part of the system's software architecture and it would be used at run-time. In this paper we suggest a modeling methodology based on the definition of the concept of *effect*. For a better understanding of the proposed methodology and for clarity reasons we rely on the UML class diagram shown on Figure 5. The idea of the methodology is to partition the model into three levels of abstraction. Level 0 would include the general concepts: effect, sensor and actuator. We suppose that everything in an ambient environment fits under these three super categories. We can make an analogy here with the concept of "Interface" in Object Oriented Programming. The designer intervenes at Level 1 where he/she declares entities that inherit from those of Level 0. Level 1 should include any entities that the designer estimates might influence the diagnosis results even if, in some cases, they are not controlled by the system, such as doors or windows. In that case, to be accurate, it would be appropriate to declare two sub-entities of actuator; one would represent controlled actuators and the other uncontrolled actuators. The latter entity will be considered as a super class for entities such as "window" or "door" if they are not controlled by the system. In Level 2 we find the different instances of the abstract entities, declared in Level 1, in a particular ambient environment.

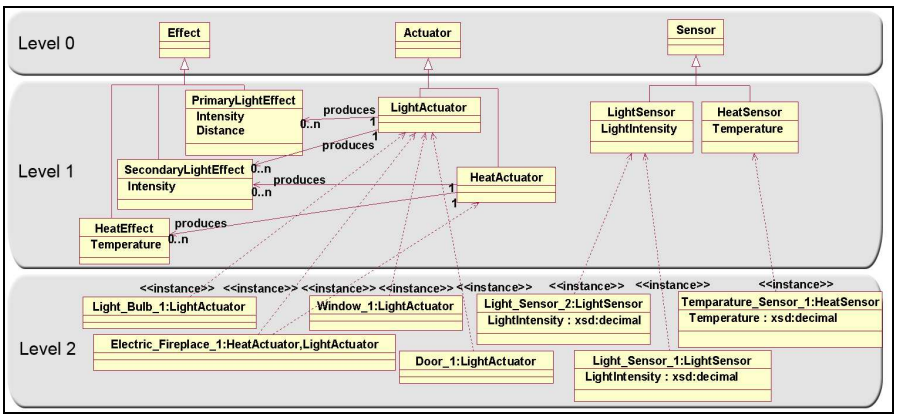

Figure 5. UML class diagram for the proposed modeling methodology applied to an example

5. Example and Scenario

To illustrate our effect-based model we present here a simple yet representative example on how the modeling process would be like for a simple example of an ambient environment. We suppose that we have an environment composed of three sensors (two light sensors and a temperature sensor),

two actuators (a light bulb and an electric fireplace) and two built-in objects (a door and a window). The first step would be to apply the modeling methodology to identify the different entities involved in the environment. The generated diagram is presented in Figure 5. The diagram uses UML notation for clarification purposes since UML offers a graphical syntax familiar to systems designers. From this diagram we can identify the main entities in the environment and the different levels of modeling. The next part is a description of the system based on the diagram and the effect-based model presented in the ontology. As presented previously, Level 0 includes only the concepts of Effect, Sensor and Actuator. In Level 1 we have the classes of entities for each of the major concepts: LightSensor and HeatSensor for sensors, LightActuator and HeatActuator for actuators, PrimaryLightEffect and SecondaryLightEffect which are effects produced by entities of type LightActuator, and finally HeatEffect which is an effect produced by entities of type HeatActuator. It is to be noted that both primary and secondary light effects possess the effect property LightIntensity which is detected by the entity LightSensor. However PrimaryLightEffect has one more property which is Distance (the distance between the source and the detector). This difference between the two light effects produced will affect the choice of the dependency law to be used. Obviously PrimaryLightEffect will use a finer dependency law. As for HeatEffect, it has the effect property Temperature which is detected by the entity HeatSensor. In Level 2 we have the instances which are the real objects in the environment: Light_Sensor_1 and Light_Sensor_2 instances of LightSensor, Temperature_Sensor_1 instance of TemperatureSensor, Light_Bulb_1 instance of LightActuator, Electric_Fireplace_1 instance of both HeatActuator and LightActuator, and finally Door_1 and Window_1 instances of LightActuator. Using the declarations in Level 1 we can deduce that Light_Sensor_1 and Light_Sensor_2 “detects” LightIntensity, Temperature_Sensor_1 “detects” Temperature, Light_Bulb_1 “produces” PrimaryLightEffect and SecondaryLightEffect, Electric_Fireplace_1 “produces” HeatEffect and SecondaryLightEffect, and Door_1 and Window_1 “produces” PrimaryLightEffect and SecondaryLightEffect.

For the diagnosis part of this scenario we use the ontology proposed in Figure 1. Each of the sensors has an instance of SensorDiagnosisProcess. In our case we would have 3 diagnosis processes: Light_Sensor_1_Diag_Process, Light_Sensor_2_Diag_Process, Temperature_Sensor_1_Diag_Process; each one deduces diagnosis from one or more DependencyLaw. In our example we estimate that we will have 3 types of DependencyLaw: heat propagation, decrease of light intensity and indirect illumination or light reflection. In this example we will illustrate numerically only the decrease of light intensity. This law is used by both light sensors’ diagnosis processes. It states that the intensity of light as a function of the distance from the light source follows an inverse square relationship. Reasoning is applied at this point, in order to find out which DependencyLaw is associated with which SensorDiagnosisProcess by linking the corresponding sensor to one or more effect properties. The latter are what constitute effects; similarly entities referenced by DependencyLaw depend on them to make calculations, hence the deduction of the concrete link between an actuator

and a sensor. Let's apply this reasoning to Light_Sensor_1, Light_Sensor_2 and Light_Bulb_1. We suppose that the latter is a 60 watt light bulb that emits 850 lm (lumen). We suppose also that it is located at 2 meters from Light_Sensor_1 and 5 meters from Light_Sensor_2 and that it is turned on. It should be noted that for the model to be adapted to the dynamic nature of ambient systems, the distance information is to be deduced automatically; this part of the model being not fully developed yet, we simply introduce the distance values manually for now. Behind every DependencyLaw there is a computational model that calculates values, using a mathematical formula, and then affects to each EffectProperty related to the DependencyLaw the estimated value. Each diagnosis process compares the value read by the related sensor with the value estimated by the DependencyLaw. If the compared values are different then the diagnosis is declared to fail and an error message is generated as explained in "Reasoning Engine" section. The computational model behind the light intensity decrease dependency law used by both sensors' diagnosis processes should return 212.5 lm for Light_Sensor_1_Diag_Process and 34 lm for Light_Sensor_2_Diag_Process. Let's say that both sensors read the same value 0 lm which means that the room is supposed to still be dark, in this case both diagnosis processes return a failed diagnosis state and the global diagnosis state should generate the proper error message asking the user to replace or to check the real state of the light bulb. The user's feedback can be added as a statement to the ontology, and can be useful for further reasoning about the light bulb. We can imagine a case in which the user's feedback confirms that the light bulb is properly illuminated even though the system says it is not; in that case the system deduces that the sensors are not functioning properly. Even though Door_1 and Window_1 are not controlled by the system they are considered as actuators, in fact they are considered in the diagnosis as anticipation for when the model will be expanded to support recovery. For now this special kind of actuators will simply appear in the error messages as a correction suggestion to the user.

6. Conclusions and Future Work

In order to improve the diagnosis of ambient systems we have introduced a new idea consisting of simulating what happens in reality by applying the appropriate physical rules in ambient environments, thanks to the concept of *effect*. Effects model the relations between sensors and actuators, based on physical laws. One of the major contributions of this work is the fact that the ad-hoc link between Actuators and Sensors is not required to be specified explicitly: actuators are not linked to sensors by the designer; the links may be calculated using effects at run-time. A local diagnosis process then compares the read value and the estimated value. Our meta-model meets the reality and particularities of ambient environments. Those environments are very dynamic and unstable in the sense that new entities may be introduced or removed at run-time, which makes their design complicated. Our work therefore proposes an original way of performing diagnosis in ambient environments. As future work, we envision different improvements to the method exposed

in this article. For instance, the issue of the possibility of having faulty sensors was explored but never really solved. A solution might be to integrate a probabilistic model to our effect-based model. Furthermore, the prospect of using different diagnosis techniques to manage distributed diagnosis of networked systems was anticipated in the model but it has not been used yet. Besides, as stated earlier, the user is the center of an ambient intelligent system, as the main purpose of the system is to satisfy his/her preferences. Yet, the user, this fundamental and really complex part of the system, is not represented in our proposed model. Indeed, contrary to the system's behavior which is predictable and thus can be easily and usefully modeled, the behavior of the user is unpredictable, which makes its modeling intricate. However explicitly modeling user behavior, tasks and needs would allow the system to provide useful assistance. Finally real-scale tests in an experimental intelligent room will be performed in order to validate the framework.

7. References

[1] Estrin D, Culler D, Pister K, Sukhatme. "Connecting the Physical World with Pervasive Networks". IEEE Pervasive Computing. Mobile and Ubiquitous Systems, 1(1):59–69, January 2002.

[2] J. Bohn, V. Coroama, M. Langheinrich, F. Mattern, M. Rohs, "Social, Economic, and Ethical Implications of Ambient Intelligence and Ubiquitous Computing". Journal of Human and Ecological Risk Assessment, Volume 10, No. 5, October 2004.

[3] JC. Augusto, P. McCullagh, V. McClelland, and J-A. Walkden. "Enhanced Healthcare Provision through Assisted Decision-Making in a Smart Home Environment". Proceedings of the 2nd Workshop on Artificial Intelligence Techniques for Ambient Intelligence, 2007.

[4] Dr. C. Kitts. "Managing space system anomalies using first principles reasoning". IEEE Robotics & Automation Magazine, Sp. Issue on Automation Science, v 13 no 4, December 2006.

[5] J de Kleer. "Focusing on Probable Diagnoses". Readings in model-based diagnosis, 1992.

[6] M. Born and E. Wolf. "Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light". Cambridge University Press, Cambridge, England, 2002, 7th ed.

[7] J. Kurien, X. Koutsoukos, and F. Zhao. "Distributed diagnosis of networked hybrid systems". AAAI Spring Symposium on Information Refinement and Revision for Decision Making: Modeling for Diagnostics, Prognostics, and Prediction, pages 37–44, Stanford, CA, March 2002.

[8] Mike Dean and Guus Schreiber. "OWL Web Ontology Language Reference". W3C Recommendation Feb. 2004.

[9] Maxwell Lewis Neal, John H. Gennari, Theo Arts & Daniel L. Cook. "Advances in Semantic Representation for Multiscale Biosimulation: A Case Study in Merging Models". Pacific Symposium on Biocomputing 14:304-315 (2009).

[10] J. Carroll et al. "Jena: Implementing the Semantic Web Recommendations". Tech. report HPL-2003-146, Hewlett Packard Labs, 2003.

[11] José R. Hilera, Francisco Ruiz. "Ontologies in Ubiquitous Computing". International Conference on Ubiquitous Computing: Applications, Technology and Social Issues, Alcalá de Henares, Madrid, Spain, June 7-9, 2006.