Proceedings of the 2015 Winter Simulation Conference
L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, eds.

MODELING AND SIMULATION OF WEB-OF-THINGS SYSTEMS PART 1: SENSOR NODES

Mircea Diaconescu
Gerd Wagner

Brandenburg University of Technology
Institute of Informatics
P. O. Box 101344
03013 Cottbus, GERMANY

ABSTRACT

In the Web of Things (WoT), special communication networks composed of sensor nodes, actuator nodes and service nodes form the basis for new types of web application systems, which are directly connected to the real world via sensors and actuators. The simulation of a WoT system allows evaluating different design options. It may require supporting "hardware in the loop", "software in the loop" and "humans in the loop". We propose several elements of a conceptual framework for simulating WoT systems at the application level, focusing on the simulation of three simple types of sensors. Our conceptual framework includes an ontology of WoT systems as sensor/actuator systems, and a meta-model for defining a WoT simulation language.

1 INTRODUCTION

In the Web of Things (WoT), special communication networks composed of sensor nodes, actuator nodes and service nodes form the basis for new types of web applications, which are physically connected to the real world. Such WoT apps can be private, such as smart home apps, personal robotics apps and factory control applications, or public, such as air pollution monitoring systems and city parking management apps. For three different reasons, many of these WoT systems (WoTS) will be based on devices with constrained resources, such as sensor nodes with low-memory microprocessors or small single-board computers running special WoT services. First, many WoTS will have to support scenarios that require battery-operated devices with little memory and low-bandwidth connections. Second, many use cases will require large-scale networks, which can only be afforded by keeping the cost for each node reasonably low. Third, there will be many personal consumer use cases, where low purchase prices, which can only be achieved by constrained resource devices with mass-manufactured standard components, are essential for establishing WoT apps.

Therefore, it is important to distinguish between WoTS that do not have the limitations implied by constrained resource devices, and WoTS based on constrained resource devices, requiring small footprint, and possibly low energy technologies.

We propose a conceptual and computational model of sensor and actuator nodes as functional components of a WoTS that physically interact with its inanimate environment, and of the WoTS as a whole communicating with web services and user agents, abstracting away from low-level networking issues, such as the network topology.

In this paper, we focus on modeling sensor nodes consisting of a controller and one or more detectors or composite sensors. The main contribution of the paper is a new general model of sensor nodes

presented in Section 3, which distinguishes between three types of detectors: analog quality detectors, digital quality detectors, and event detectors. This model forms an essential part of our *WoTS ontology*. In Section 3.1, the physical interaction of detectors with their environment is modeled with the help of the following transformation functions:

1. *quality-to-voltage* and *voltage-to-quantity* for analog quality detectors,
2. *quality-to-bytes* and *bytes-to-quantity* for digital quality detectors,
3. *externalEvent-to-sensorEvent* and *sensorEvent-to-internalEvent* for event detectors.

In Section 4, based on our WoTS ontology, we propose a meta-model as a basis for defining a WoTS simulation language, and a design model for the core of a WoTS simulator.

2 RELATED WORK

In (Karnouskos and Tariq 2008), an approach to agent-based simulation of a network of web-service-enabled devices is proposed. The authors argue that the *Service-Oriented Architecture* (SOA) paradigm can be used for achieving interoperability between the nodes of a WoT system and between such a system and modern enterprise networks. When devices expose their data and operations as web services, this provides an integration of devices with enterprise applications, allowing new innovative solutions to enterprise automation problems. The authors choose the open standard protocol *Devices Profile for Web Services* (DPWS) as the interaction protocol for web-service-enabled devices (Microsoft 2006). However, this protocol is based on a protocol stack that is too complex for constrained resource devices as needed, for instance, in battery-operated WoT networks, while it may work well in non-constrained office or factory environments. A Java-based multi-agent platform is used to create the agents that simulate DPWS devices. These agents are connected to the enterprise network in the same way as real DPWS devices. The resulting system can be used for evaluating the impact of a large number of networked devices on the running enterprise applications, without the need to set up a real device network, which would be more expensive and more difficult.

In (Österlind 2006), *COOJA*, a simulator for networked small devices with limited resources (i.e., about 10 KB RAM and 30 KB flash storage) running the *Contiki* operating system, supporting many important networking technologies (such as IPv4, IPv6, TCP, UDP, and CoAP), is described. COOJA allows to create node types for which it loads and compiles the Contiki source code. All the nodes of the same type share a Contiki core, but use separate memory, network and I/O control. The simulator is based on Java and uses JNI to interact with Contiki and external C/C++ code. However, due to the resource requirements of Contiki, COOJA cannot be used for simulating typical sensor nodes, which are based on constrained memory/storage microcontrollers such as the *Arduino UNO* or the more powerful *Arduino MEGA* (Arduino Foundation 2005).

In (Gschwandtner et al 2011), a model and a software solution for the simulation of a special class of sensors, called *Light Detection and Ranging (LIDAR)* devices, or *range scanners*, integrated with the popular 3D modeling tool *Blender*, is presented.

While in (Karnouskos and Tariq 2008) a special distributed system architecture with web-service-based communication is considered, (Österlind 2006) is mainly concerned with simulating the networking infrastructure components of a IoT system, and (Gschwandtner et al 2011) focus on deeply modeling a special class of sensors used in robotics. All three works do not consider any general model of sensing and sensor nodes, nor do they propose any general WoTS simulation language or WoTS simulator.

3 IoT AND WoT SYSTEMS

An *Internet-of-Things (IoT) system* is a communication network consisting of sensor nodes, actuator nodes and service nodes, such that at least one node is connected to the Internet. A sensor node consists of a controller to which one or more sensors and a communication unit are attached. An actuator node

consists of a controller to which one or more actuators, zero or more sensors and a communication unit are attached.

A ***WoT system (WoTS)*** is an IoT system that is built with web technologies. These technologies do not only include the classical web technologies HTTP(S), HTML, CSS and JavaScript, but also the more recent web technologies *Server-Sent Events*, *Web Sockets*, and the *Constrained Application Protocol (CoAP)* proposed in (Shelby, Hartke and Bormann 2014). We distinguish between the following three cases:

1. WoT systems that do not have the limitations implied by constrained resource devices. These systems can use ordinary networking and web technologies such as IEEE 802.11 for wireless networking, HTTPS and SOAP for application-level messaging, and SOAP-based co-ordination and security techniques, as proposed in (Karnouskos and Tariq 2008).
2. WoT systems based on constrained resource devices having unlimited power supply (not using batteries), such that power consumption is not a concern. These systems need an alternative software/technology stack that is adapted to the limited main memory, storage and processor speed of the constrained resource devices. Ethernet (or IEEE 802.11) can still be used for (wireless) networking, but only CoAP or an HTTP subset, and no HTTPS, can be used for application-level messaging.
3. WoT systems based on constrained resource devices that are battery-powered, requiring low-energy wireless networking technologies, such as *IEEE 802.15.4*, and small footprint software technologies, such as CoAP for application-level messaging. These systems often have higher packet error rates and a lower throughput (say, of only tens of kbit/s).

Unlike many other authors, such as (Karnouskos and Tariq 2008, Brambilla et al 2014), we consider the issue of using the new *Internet Protocol (IP)* version 6 (IPv6) instead of the established version 4 (IPv4) as orthogonal to the WoT. The main issue solved by IPv6, allowing a greater address space than IPv4, is not necessarily an issue for WoT systems, which can, in many cases, be built with either of them. Of course, the increasing use of IoT apps will contribute to the increasing demand for IP addresses. But since most IoT/WoT devices will not have to be reachable via an IP address, the expected explosive growth of the IoT/WoT will not imply a similar explosion of the IP address space.

The following are considered to be desirable features of a WoTS:

- self-configuration: the dynamic composition of WoTS by nodes joining and leaving the network at any time
- self-diagnosis: automatic discovery of failures and faults
- self-optimization of constrained energy (battery-based) WoTS: automatic monitoring and on/off-time control of resources

3.1 The Environment

The environment of the sensors and actuators of a WoTS consists of amounts of matter (such as soil and air) and of discrete material objects (such as cars and animals). Amounts of matter and material objects bear certain physical qualities (such as color or temperature) that can be measured by quality detectors, and may participate in certain events that can be detected by event detectors.

3.2 Detectors, Sensors and Sensor Nodes

As described visually in the UML class diagram shown in Figure 1 below, a sensor node consists of a controller with a communication unit (such as the ESP8266 Wi-Fi module) and one or more sensors. A sensor consists of one or more detectors, which are simple (non-composite) sensors.

As a component of a sensor node, a sensor is a measurement device that is attached to a controller within a sensor node. For instance, a *DHT22* temperature and humidity sensor may be attached to an *Arduino UNO* micro-controller, or a *Proximity Infra-Red (PIR)* sensor may be attached to a *Raspberry Pi* single-board computer.

Notice that in the class diagram of Figure 1, the `Arduino UNO` class represents, like a product type, a controller type, the instances of which are individual Arduino UNO micro-controllers. Likewise, the `DHT22` class represents a sensor type, the instances of which are individual DHT22 sensors.

We distinguish between three types of detectors:

1. quality detectors with an analog interface,
2. quality detectors with a digital interface,
3. event detectors.

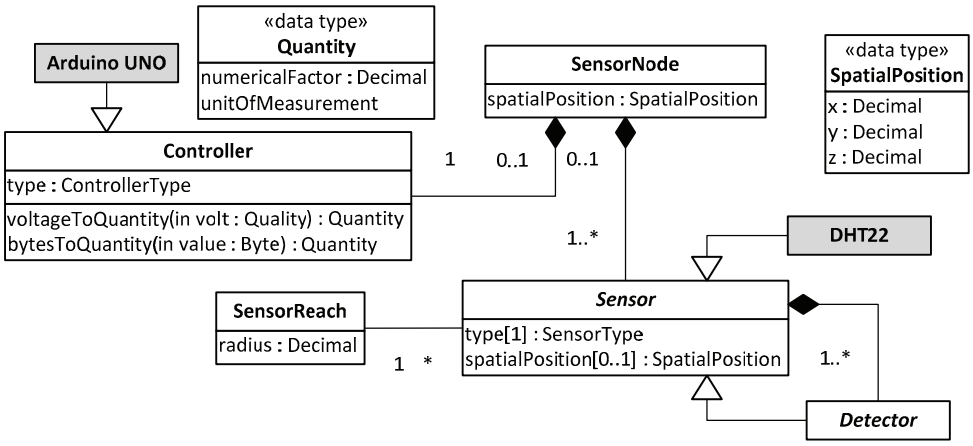

Figure 1: A fragment of a WoTS ontology describing sensor nodes.

As depicted in Figure 2 below, quality detectors and event detectors are simple (non-composite) sensors. A ***quality detector*** is a device that allows measuring a physical quality of its environment (or *reach*). An ***event detector*** allows detecting events occurring in its reach. Notice that the concept of physical qualities has been defined in the philosophical discipline of ***ontology*** (or *metaphysics*). A quality is an entity, and not a data value, but it can be approximately represented by a data value (namely the value of an attribute that captures the type of quality). For instance, the voltage level of a wire of a particular detector at some moment in time is a quality, which can be approximately represented by the value of an attribute `outputVoltage` used for expressing statements about, and measurements of, the detector.

The sensing operation of an analog quality detector can be conceptualized as a transformation, which converts a quality to be measured in the detector's reach to an internal quality of the detector device (typically, to a voltage level) that can be read and transformed to a measurement quantity by the controller. In our WoTS sensor ontology, we call the first transformation function ***quality-to-voltage***, and the second one ***voltage-to-quantity***. The sensor ontology of Figure 2 below enforces that any analog quality detector type, such as `Grove` or `LM35`, has a specific quality-to-voltage function by making any such detector type a subclass of the abstract class `AnalogQualityDetector`, such that it has to implement its abstract `qualityToVoltage` method.

The sensing operation of a digital quality detector can be conceptualized as a transformation, which converts a quality to be measured in the detector's reach to a sequence of bytes that can be read and

transformed to a measurement quantity by the controller. In our WoTS sensor ontology, we call the first transformation function ***quality-to-bytes***, and the second one ***bytes-to-quantity***.

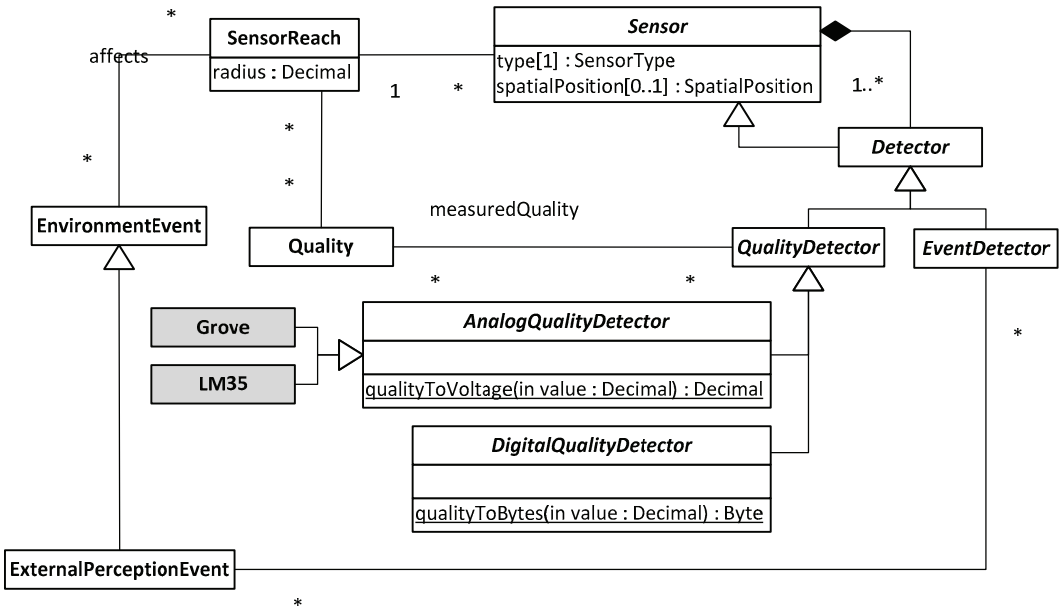

Figure 2: A fragment of a WoTS ontology describing sensors.

The sensing operation of an event sensor can be conceptualized as a transformation, which turns the occurrence of an event of a certain type in its reach to an internal event of the sensor device itself (typically corresponding to digital voltage signals) that can be detected and transformed to a control event by the controller. In our WoTS sensor ontology, we call the first transformation ***externalEvent-to-sensorEvent***, and the second one ***sensorEvent-to-internalEvent***.

3.2.1 Quality Detectors with an Analog Interface

These sensors provide a voltage level as their output. The interpretation of voltage levels by the controller involves two steps: first a digitization of the voltage level resulting in a decimal voltage number, and second a mapping of this number to a quantity data value composed of a decimal number and a unit of measure.

For example, the LM35 analog temperature sensor provides a voltage level on its output pin where 10 mV (millivolts) correspond to one degree Celsius. Using an Arduino UNO microcontroller and one of its Analog-to-Digital Convertor (ADC) input pins, the analog voltage level can be converted to a temperature quantity value in two steps. First, the voltage level is mapped to a decimal voltage number. Since the ADC is 10 bits wide and the maximum voltage level is 5 volts, the voltage range [0,5] is divided into 1024 parts (corresponding to 10 bits), where each part corresponds to about 5 millivolts, which in turn correspond to 0.5 degree Celsius.

The maximal error of the output value of a quality measurement based on a quality detector with an analog interface is obtained by combining its accuracy with the controller's maximal ADC error.

3.2.2 Quality Detectors with a Digital Interface

Sensors using a digital data interface (such as UART/USART, I2C, SPI, or 1-Wire) provide the measurement quantities in the form of binary encoded numbers to the controller. The controller interprets the binary data according to the interpretation described in the sensor datasheet.

For example, the DHT22 temperature and humidity sensor comes with a custom 1-Wire interface that provides a sequence of digital high voltage signals, where a short duration is interpreted as 0 and a long duration as 1. It sends a stream of forty such signals, containing both the temperature and humidity values. Another example is the DS18B20 temperature sensor, which comes with the standard 1-Wire interface, so its digital voltage signals can be interpreted by a controller with built-in hardware support for this interface.

Compared to analog sensors, digital sensors typically come with a higher accuracy, a higher power consumption and at a higher price.

3.2.3 Event Detectors

An event detector for detecting events of a certain type provides only a high or low digital signal when it has detected the occurrence of an event of that type. For example, a PIR (Proximity Infra-Red) sensor has its output pin going high whenever it detects an infra-red emitting object (e.g., a human or a medium to large sized animal) in its reach. This allows for example to program an asynchronous interrupt in the Arduino UNO, which triggers some action when the sensor reports an event.

A PIR sensor also includes an adjustable light sensor (so the PIR sensing is only performed at some level of light, e.g., only in the night) and an adjustable amount of time for which it remains high after detecting an infra-red emitting object. The adjustments are made by using the built-in potentiometers (adjustable resistors).

3.3 Actuators and Actuator Nodes

As a component of an actuator node, an actuator is an enactment device that is attached to a controller within an actuator node. Common types of actuators are electro-mechanical devices that are controlled with the help of a (voltage, current or digital interface) signal. Examples are motors, electro-valves and relays. We plan to treat actuators and actuator nodes in a follow-up paper.

4 MODELING AND SIMULATION OF WoT SYSTEMS

A ***WoTS simulation*** consists of one or more simulated WoTS nodes, an environment simulator, and zero or more real WoTS nodes, satisfying certain conditions as defined below. The environment simulator is in charge of managing the state of the simulated environment, including the reaches of all simulated sensor and actuator nodes, and of simulating environment events, which may change the state of the simulated environment, e.g., by changing certain property values in certain simulated sensor reaches, and may affect event detectors if a simulated environment event of the right type occurs in the reach of a simulated event detector.

We define the basic elements of a WoTS simulation language in the form of *meta-types* defined in a meta-model (or language model) represented as a UML class diagram. These meta-types (e.g., `DetectorType`) represent the concepts of the defined language. They are instantiated by the types defined in a simulation model (such as the sensor type `LM35`). The meta-model shown in Figure 3 below defines the language concepts *controller type*, *sensor type* and *detector type* with the two sub-concepts *quality detector type* and *event detector type*, which are discussed in the following subsections.

Notice that in addition to *precision* and *accuracy*, a quality detector type also defines a *resolution* representing the smallest change that can be detected in the values of the attribute that captures the quality to be measured.

4.1 Simulating the Environment

The environment of the sensors and actuators of a WoTS is simulated with the help of an *environment simulator* that represents all relevant amounts of matter and all relevant material objects from the environment, and simulates their history by applying state changes of objects and amounts of matter triggered by events.

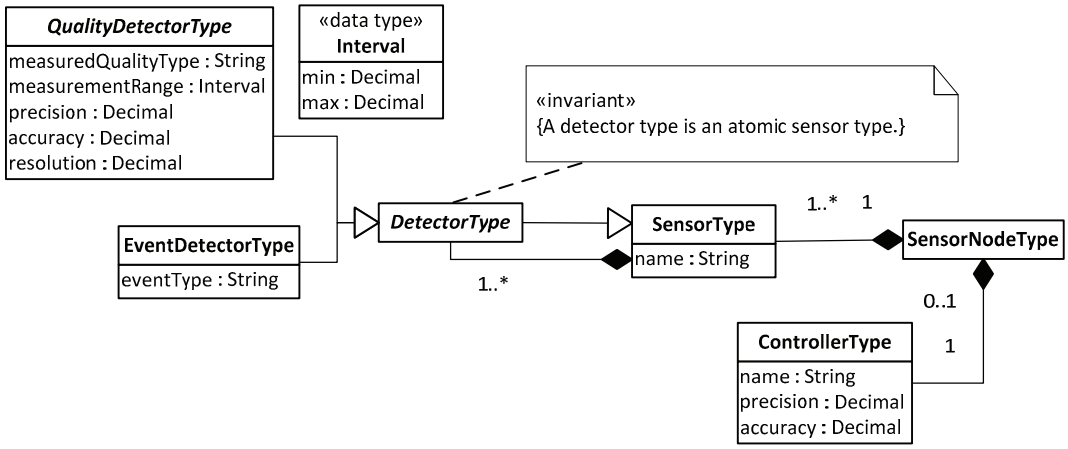

Figure 3: A meta-model defining basic language concepts.

4.2 Modeling and Simulation of Sensor Nodes

As explained in the previous section, a sensor node consists of a controller to which one or more sensors and a communication unit are attached. Since our simulation framework is not concerned with low-level communication issues, we do not include the communication unit as an explicit component of a sensor node, but only a controller and one or more sensors.

4.2.1 Modeling and Simulation of Sensors

As depicted in the diagram shown in Figure 4 below, the simulated reach of an analog quality sensor/detector is described with the help of attributes, which capture quality types, such that at any moment in time, the reach has a specific value for each of its attributes in a slot.

4.2.1.1 The *Attribute-Value-to-Voltage-Value* Function

In a measurement simulation, the value of a reach attribute is read by the simulated quality detector and transformed to a voltage value with the help of the `attrValToVoltVal` function defined in the class representing the detector type. The voltage value is then processed by the simulated controller and mapped to a measurement quantity by applying the `voltValToQuantity` function defined in the class representing the controller type. The attribute-value-to-voltage-value function maps the attribute value to a voltage value in three steps:

1. In the first step, the sensor resolution (as specified in its datasheet) is taken into consideration. Two rounding methods are used for computing the sensor's input value from the measured attribute value: *ceil* and *floor*, both of them being based on a significance factor, which is equal to the sensor resolution.

2. In the second step, the sensor's input value is mapped to a voltage value based on the sensor's electronic design. In some cases, the function is described in the sensor's datasheet, so it can be implemented in the class representing the sensor type. Our goal is to support many widely used sensors in the simulation framework by providing pre-defined sensor types for them.
3. In the third step, the sensor's error characteristics, i.e., precision and accuracy, as provided in the sensor's datasheet, are taken into consideration.

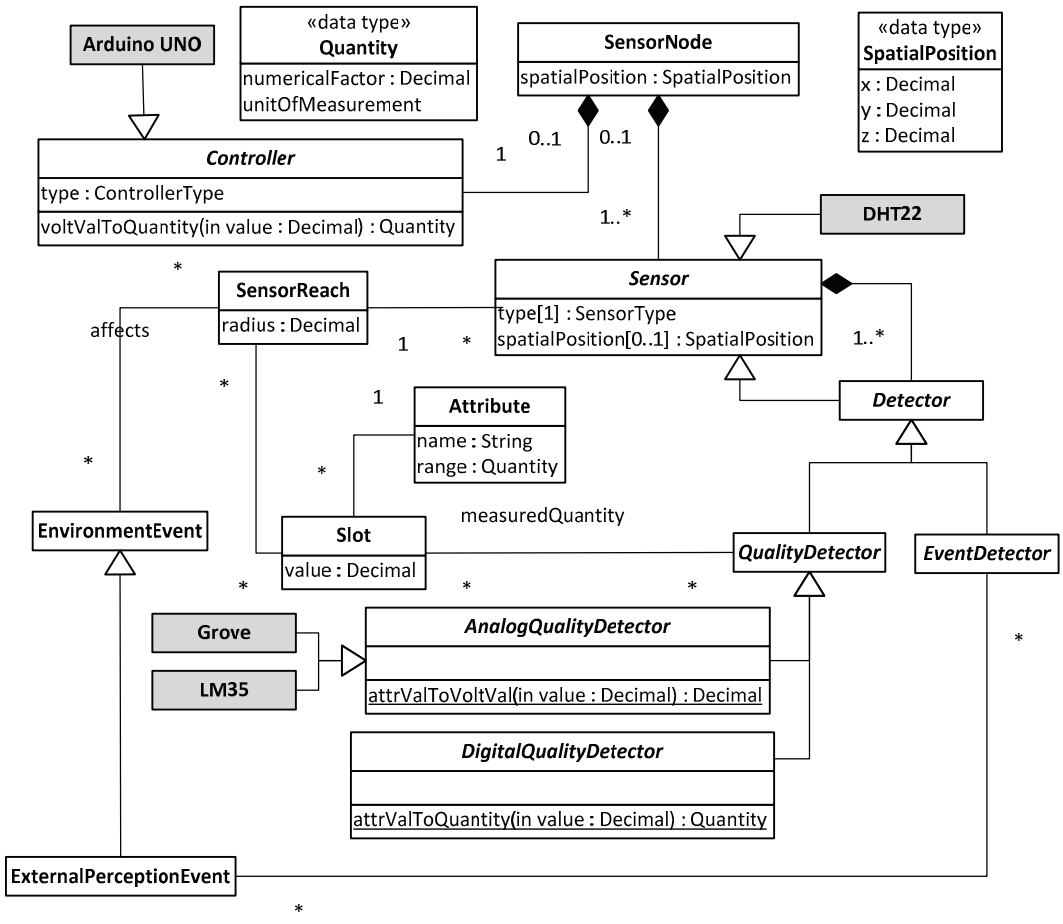

Figure 4: A design model defining generic classes of the WoTS simulator.

For example, a Grove moisture sensor has a resolution equal to 1. For a simulated reach moisture value of 27.38, applying a ceil mapping with a significance factor of 1, the sensor input moisture value is 28, while applying a floor mapping with the same significance factor, the sensor input moisture value is 27. Notice, that the measurement unit of the measured quality does not matter for the sensor resolution.

When the sensor datasheet does not describe how the sensor's input value is mapped to a voltage value, we can still use a linear, logarithmic or exponential function as an approximation in many cases. Since the sensor output values for the minimum and the maximum values of the measured quality may be obtained or derived from the sensor datasheet, i.e., (q_1, v_1) and (q_2, v_2), the following functions are obtained:

- Linear: $f(x) = \left(\frac{v_2 - v_1}{q_2 - q_1}\right)(x - q_1) + v_1$, derived from $f(x) = ax + b$.

- Logarithmic: $f(x) = \frac{v_1 - v_2}{\ln(\frac{q_1}{q_2})}(\frac{v_2 \ln(q_1) - v_1 \ln(q_2)}{v_1 - v_2} + \ln(x))$, derived from $f(x) = aln(bx)$.
- Exponential: $f(x) = v_1(\frac{v_1}{v_2})^{\frac{x-q_1}{q_1-q_2}}$, derived from $f(x) = ae^{bx}$.

For example, the LM35DZ analog temperature sensor, which is a variant of LM35 measuring temperature in the range 0-100°C, can be used with the linear mapping function. The known output voltage values computed on the basis of the sensor datasheet are: 0V for 0°C and 1V for 100°C, i.e., $q_1 = 0, v_1 = 0, q_2 = 100, v_2 = 1$ (the sensor output is 10mV for each degree Celsius). As a result, the used linear mapping function is: $f(x) = 0.01x$, where $f(x)$ is the internal sensor voltage in volts and x is the measured temperature value in degree Celsius.

The last step required for getting the final sensor output is to consider its accuracy and precision error factors. In (BIPM 2012), accuracy is defined as the closeness of agreement between a measured value and a true value, while the precision is defined as the closeness of agreement between indications or measured values obtained by replicate measurements of the same or similar objects under specified conditions. These characteristics may be approximated by using a normal distribution function defined by a *mean* and a *standard deviation*. In the case of accuracy, the mean is equal to the middle of the accuracy range (e.g., 0 for the LM35DZ temperature sensor which has an accuracy of ±0.5°C) while the standard deviation is one third of the accuracy interval (e.g., 0.3°C in the case of the LM35DZ sensor). The obtained accuracy error value is then added to the internal voltage value to produce the output voltage.

The precision of a sensor is often not specified in the sensor datasheet. When this information is available, it can also be approximated by using a normal distribution. For this, the sensor needs to keep a log of its first accuracy error value for a specific measured quality, which represents the mean of the normal distribution. The standard deviation is approximated as one third of the precision value.

Since precision mostly depends on the state of the sensor's reach when the measurement is performed, for a better approximation, it is recommended to keep logs of the entire reach states rather than only the accuracy error. This allows to have a more exact simulation of precision, but it also significantly increases the resources required for the simulation execution (i.e., used memory and execution time). Therefore, it should be an option to simulate precision in one of the two ways presented above.

4.2.1.2 The *Voltage-Value-to-Quantity* Mapping Function

Using an ADC-enabled controller (like the Arduino UNO) and simple mathematical formulas, the output voltage of an analog quality sensor can be translated to a decimal number representing the quantity of the measured quality. The output of an ADC unit is represented as a readable registry, with a maximum number of binary bits, as specified in the datasheet (e.g., 10 bits for Arduino UNO). The input analog voltage is transformed to an integer number in the interval $[0, 2^n - 1]$, where n represents the maximum number of bits of the ADC registry. This representation allows to compute the ADC resolution using the following formula: $r = \frac{maxV - minV}{2^n}$, where $maxV$ and $minV$ represents the maximum and minimum input voltage value accepted by the ADC controller. The ADC bits representation is mapped to a decimal number using the following formula: $v = r\, v_{adc}$, where r is the ADC resolution and v_{adc} is the integer number read from the ADC registry.

The ADC unit has accuracy and precision, likes a sensor, since it is a voltage sensor with a digital output. Using the same approach as in the case of the analog sensors, the accuracy and precision error factors are reflected in the ADC output.

An analog sensor is used together with an ADC-enabled controller, and both of them have error characteristics, i.e., resolution, accuracy and precision. The final output quantity reflects the error for both components of the sensor node.

4.2.1.3 The *Attribute-Value-to-Quantity* Mapping Function

This case applies to sensors with digital output, no matter which communication protocol they use, since the final result is a decimal number. When the sensor datasheet specifies a particular mapping function, then it can be used in the sensor simulation. However, in many cases, this information is not available, and then the sensor input to output behavior may be represented as a linear, logarithmic or exponential function. The strategy is to use the same mapping functions as in the case of the attribute-value-to-voltage-value function, with the difference that the output is not a voltage value but a value which is the quantity of the measured quality.

The sensors with digital output have also accuracy, precision and resolution factors, which are handled in a very similar way as in the case of the attribute-value-to-voltage-value function.

4.3 Modeling and Simulation of a WoTS as a Whole

A ***WoTS simulation*** consists of one or more simulated WoTS nodes, an environment simulator, and zero or more real WoTS nodes, such that

1. All simulated quality detectors (on simulated sensor and actuator nodes) can sense/read as their input the value of an attribute of their simulated reach corresponding to the quality to be measured. A simulated analog quality detector first maps the attribute value to a voltage value with the help of an attribute-value-to-voltage-value function, such that the simulated sensor node can then map it to the simulated measurement result with the help of a voltage-value-to-quantity function. A simulated digital quality detector directly maps the property value to the simulated measurement result with the help of an attribute-value -to-quantity function.
2. All simulated event detectors can detect simulated external perception events as inputs from the environment simulator
3. All simulated actuators on simulated actuator nodes create simulated action events as outputs to the environment simulator, which maps them to simulated physical signals as inputs to simulated sensors in the reach of the simulated actuator.
4. Simulated sensor nodes are not "coupled" with real actuator nodes: the reach of any real actuator node does not overlap with the reach of any simulated sensor node. The reach of an actuator node is its local environment, in which real state changes can be caused by it. The reach of a simulated sensor node is the spatial region corresponding to its sensing radius in the simulated local environment of the simulated sensor node.

This definition of a WoTS simulation includes the special case where all nodes are simulated.

Notice that while there is a crisp boundary between real and simulated sensor and actuator nodes, the boundary between real and simulated service nodes is more fuzzy, since service nodes are not connected to the "real world", but only to digital network signals, which may represent real or simulated signals.

5 A GREEN HOUSE TEST CASE

Our Green House test case considers a closed environment with three important parameters: soil moisture, air temperature and relative humidity. A generic plant type is considered and the quantity of water consumption per unit of time for this plant is known, and this, together with the temperature variable (which causes water to vaporize) affects the soil moisture level as well as the relative humidity.

5.1 Simulated Hardware Configuration

Three sensors are simulated, and they correspond to the three environment parameters:

- DHT22/AM2302 digital temperature and humidity sensor with a custom 1-Wire digital interface (two quality detectors on the same physical package, with a common data interface). It allows measurements of air humidity in the range from [0, 99]% with a typical accuracy of ±2% and precision of ±1%, as well as temperature measurements in the range from [-40, 80]°C with a typical accuracy of ±0.5°C and precision of ±0.2°C.
- GROVE analog soil moisture sensor, with a range of 0 – 100% and a typical accuracy of 10%. The datasheet does not provide information about the sensor's precision, but our research on the web has shown that typically a value of ±5% is to be expected for this sensor type. This sensor has an analog quality detector.

In general, the goal is to allow using real devices in combination with simulated devices. Technically, this is possible by using a virtual router, a piece of software, which allows the simulated sensor nodes to behave in the network same as real hardware: connects to a Wi-Fi or Wired network, acquire IP from DHCP and use standard communication protocols, e.g., CoAP over UDP, for data transmission.

5.2 Environment Simulation

The environment of our scenario, where the simulated sensors exists, is also simulated. The variation of the air temperature, air humidity and soil moisture is perceived by the simulated sensors and transformed to quantities, e.g., degrees Celsius for temperature.

The environment space is defined as discrete, composed of quadratic cells. A cell represents the smallest and indivisible space unit. A reach, where a sensor exists, represents a neighborhood set of such cells. The environment simulator takes into consideration the following parameters:

- *Temperature*: the variation from day to night is considered linear. The variation interval is set for 24 hours. The temperature starts to increase after sunrise until a specified daytime, e.g., 4:00PM for summer time, then decreases until the next sunrise. Random variations may occur, with a specified periodicity, allowing to simulate close to reality events, such as wind and clouds. For this scenario, the temperature distribution over the simulated environment is considered uniform, that is, the temperature value is the same no matter the coordinates in the reach.
- *Soil moisture*: the variation depends on both, the temperature (affects the water vaporization), as well as the quantity of water known to be consumed by the plant. When the watering occurs, the water distribution in soil is computed by using the following formula: $\frac{d\theta}{dt} = \frac{d}{dr}\left(D\frac{d\theta}{dr}\right) - S$, where θ is the volumetric water content, t is the time, D is the soil water diffusivity, r is the radius, and S is the water uptake by the plant(s) root.
- *Air humidity*: the variation depends on the temperature and water dew point. The formula used to compute the relative air humidity is: $RH = (10^{m[\frac{T_d}{T_d+T_n} - \frac{T_a}{T_a+T_n}]}) * 100\%$, where T_d is the dew point temperature, T_{at} is the temperature in the environment, while m and T_n are constants which depends on temperature ranges and are provided in constant tables.

In general, for a specific simulation scenario, additional specifications can be provided by the scenario author, these coming as an addition or as a complete replacement of the framework capabilities.

For simulations with sensor hardware in the loop, it is important to notice that while simulated sensors have a simulated reach, real sensors are situated in a real-world environment. A simulated environment does not affect any real sensor node and a real-world environment does not affect any simulated sensor node, but the two can be part of the same sensor network, share the same gateway and provide data to the same services.

6 CONCLUSIONS

We have presented a proposal for modeling and simulating certain types of sensor-actuator systems and WoT systems consisting of simple sensors based on quality detectors and event detectors, such as LM35 analog temperature sensors and *Proximity Infra-Red (PIR)* sensors. Although our approach is more general than the approaches discussed in the section on related work, it does not provide a completely general model of sensors, since it does, for instance, not account for more advanced types of sensors such as LIDAR devices and video cameras.

We plan to present a proposal for modeling and simulating the actuator nodes and the energy consumption of a WoTS in a follow-up paper. We also started to implement our proposal and expect to be able to present evaluation results in the follow-up paper.

REFERENCES

Arduino Foundation. 2005. "Arduino Platform". http://arduino.cc.

BIPM 2012. "International Vocabulary of Metrology – Basic and General Concepts and Associated Terms." VIM 3rd edition. JCGM 200:2012. http://www.bipm.org/utils/common/documents/jcgm/JCGM_200_2012.pdf

Gschwandtner, M., R. Kwitt, and A. Uhl. 2011. "BlenSor: Blender Sensor Simulation Toolbox." *Advances in Visual Computing, 7th International Symposium (ISVC 2011)*, Las Vegas, Nevada, USA.

Brambilla, G., M. Picone, S. Cirani, M. Amoretti and F. Zanichelli. 2014. "A Simulation Platform for Large-Scale Internet of Things Scenarios in Urban Environments." In *Proceeding of the First International Conference on IoT in Urban Space (Urb-IoT 2014)*, 50-55. Rome, Italy: Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (ICST).

Karnouskos S., and M. M. J. Tariq. 2008. "An Agent-based Simulation of SOA-ready Devices." In *Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, 330-335. IEEE Computer Society.

Microsoft Corporation. 2006. "Devices Profile For Web Services." Accessed March 19, 2015. http://specs.xmlsoap.org/ws/2006/02/devprof/devicesprofile.pdf.

Österlind, F. 2006. "A Sensor Network Simulator for the Contiki OS." SICS Technical Report. ISSN: 1100-3154. ISRN:SICS-T-2006/05-SE.

Shelby, Z., K. Hartke, and C. Bormann. 2014. "Constrained Application Protocol (CoAP)". RFC 7252, June 2014, Internet Engineering Task Force (IETF).

AUTHOR BIOGRAPHIES

MIRCEA DIACONESCU is an Academic Employee and Ph.D. student at the Chair of Internet Technology, Brandenburg University of Technology, Germany. He holds B.Sc. and M.Sc. degrees in Computer Science from Craiova University, Romania. His main research areas are simulation technologies and the Web of Things. His email address is M.Diaconescu@b-tu.de.

GERD WAGNER is Professor of Internet Technology at Brandenburg University of Technology, Germany. His research interests include (agent-based) modeling and simulation, foundational ontologies, knowledge representation and web engineering. His email address is G.Wagner@b-tu.de.