75 Fifth Street NW, Suite 312
Atlanta, GA 30308, USA
voice: +1-404-592-6897
web: www.InterCAX.com
email: info@InterCAX.com

Author: Dr. Dirk Zwemer, dirk.zwemer@intercax.com

Date: Dec 20, 2015

Technote: Smart Home – Modeling the Internet-of-Things with SysML

Part 2 Functional Design to Procurement

Contents

Abstract

The objective of this series of tech notes is to illustrate a complete process for specifying, designing and evaluating a representative Internet-of-Things Smart Home model using Model-Based Systems Engineering and SysML. Part 2 demonstrates the structural modeling of a reference architecture with traceability back to the functional model, and the use of SysML parametrics to create the capability of evaluating cost and power consumption directly from the model itself. It also covers the first step in configuring and instantiating the real design, using Syndeia to link to IoT component product data in an external database.

Introduction

The premise of this series is that Model-Based Systems Engineering (MBSE) using a SysML modeling tool is a powerful approach to the design and implementation of real Internet-of-Things (IoT) systems. This power is based on a number of features of SysML, including

- Scope – SysML provides a common language and workspace for multiple aspects of MBSE, including requirements, behavior, structure and analysis. It also spans multiple technical domains: mechanical, electrical, software, etc.
- Traceability – SysML provides explicit mechanisms managing connections between models. In our example, we use this for internal links inside the SysML model (e.g. <satisfy> dependencies between requirements and functions in Part 1, <allocate> dependencies between function and structure in Part 2. We also use this for "external" links between the SysML model and MySQL repositories of product data.
- Executability – The precise semantics of SysML enable automated query and analysis of the model itself, without the normal disconnect between modeling and simulation. This pays off especially for frequent, repetitive analysis, including requirements verification and evaluation of alternative designs. We will use this for parametric evaluation of cost and power consumption for our Smart Home.

In the first note in this series, we described an MBSE process going from requirements specification to a detailed multi-level functional design. In this, the second of the series, we will

- define a structure for the system, initially with a reference architecture independent of the specific home or IoT product choices.
- check that the functions fully map to the structure, both at the component and interface levels.
- add parametric calculations to the reference model that will apply to multiple configurations.
- draw IoT device product data from a MySQL database into the SysML model.

In the third note, we will create a specific configuration, a four-room home, checking functionality against the reference model (functionality and requirements), and evaluate cost and power consumption using the embedded parametric models. The final note in the series describes how this model can link to both smaller models, e.g. CAD models of individual components, and larger models, e.g. the Smart Grid for electrical power.

Creating a Structural Model

Based on the functional model, we begin with the structure shown in Figure 1.

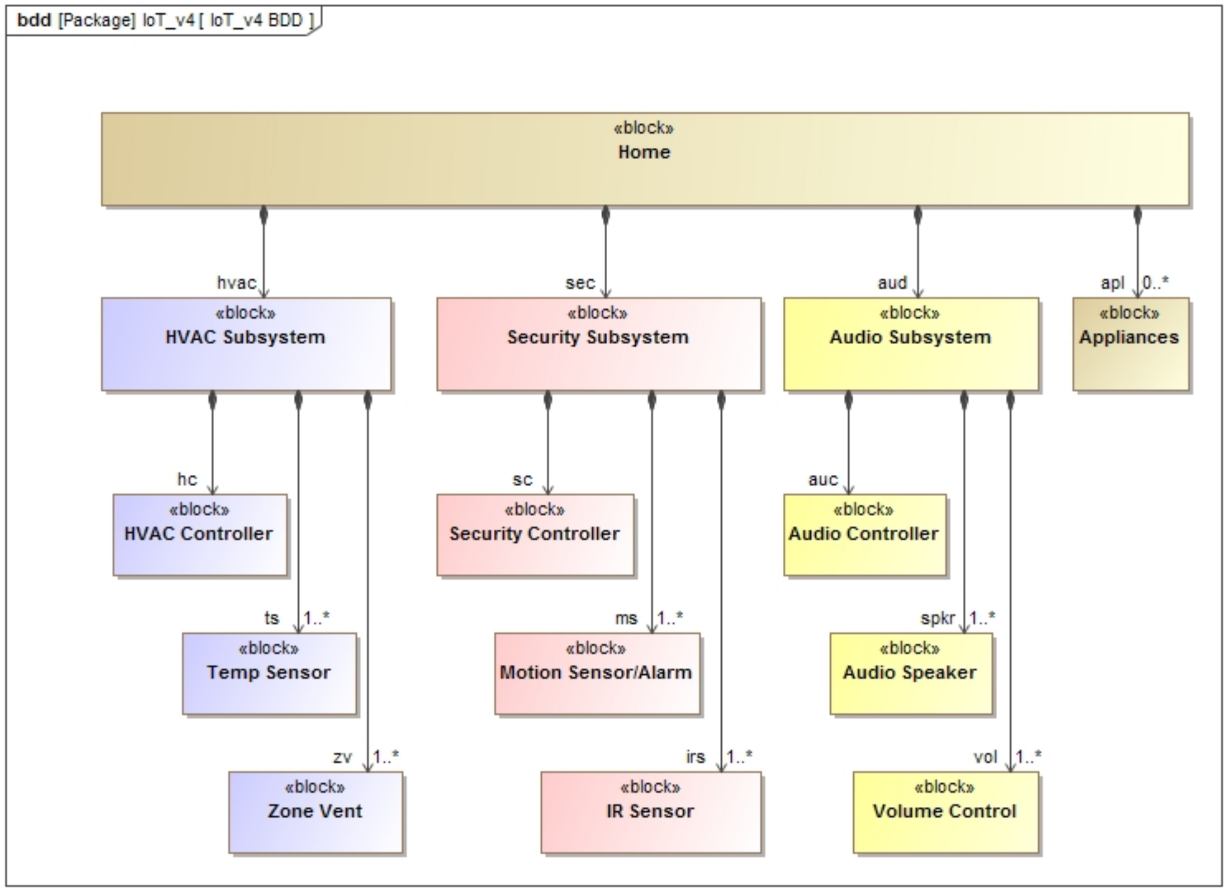


Figure 1 Smart Home Reference Model - Structure

Each primary functional domain is assigned to a different subsystem, as shown in Figure 2.

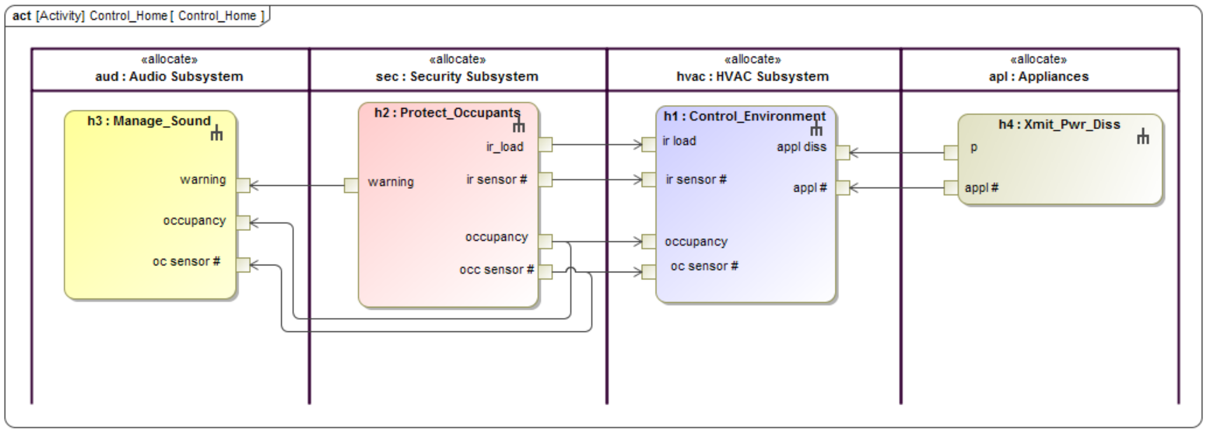


Figure 2 Top-level Activity diagram for Control Home activity with swimlane assignment to structure

Each of the second level activities can be further decomposed and allocated to the parts of each subsystem. For example, we propose to build the security subsystem with a single central controller, one-to-many infra-red (IR) sensors to detect heat sources and one-to-many motion sensors to detect occupancy. The activities making up the Protect Occupant activity are individually allocated to the parts, as shown in Figure 3. Note that both the sec3:Sound Warning and the sec2:Detect Occupancy actions are assigned to the Motion Sensor/Alarm, describing a very specific combination of features for this component. Similar allocations are made for the other subsystems.

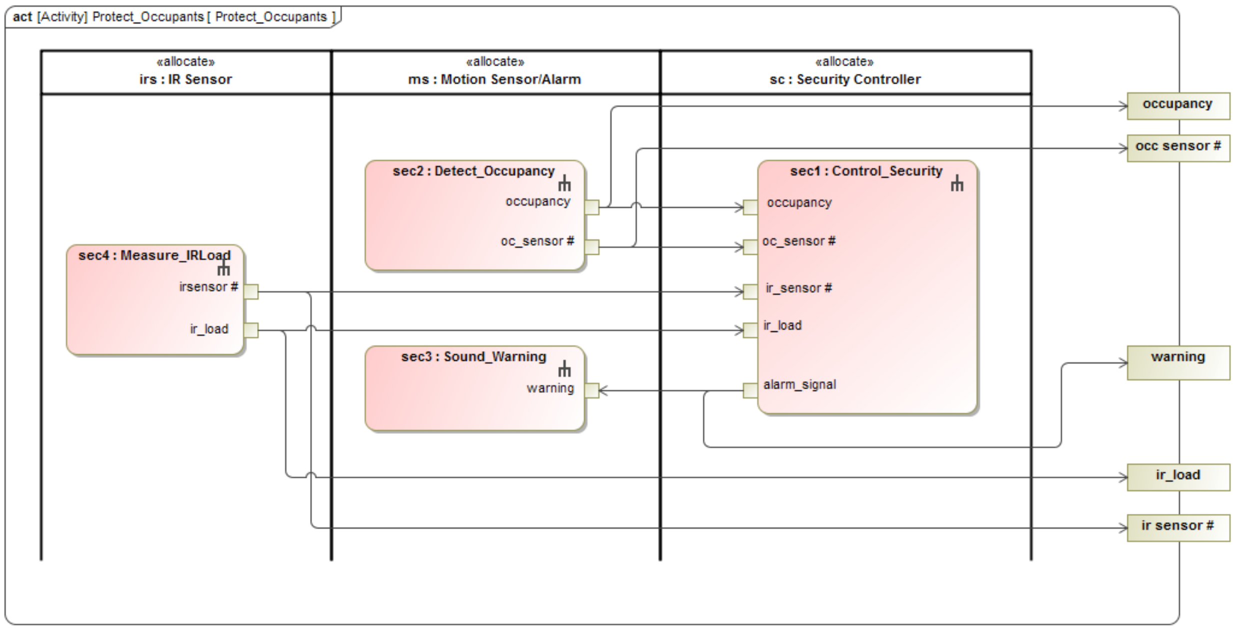


Figure 3 Second-level Activity diagram for Protect Occupant (Security), with structural allocations

The other aspect of structural description is connectivity, which can be shown on an internal block diagram for the Home blocks in Figure 4.

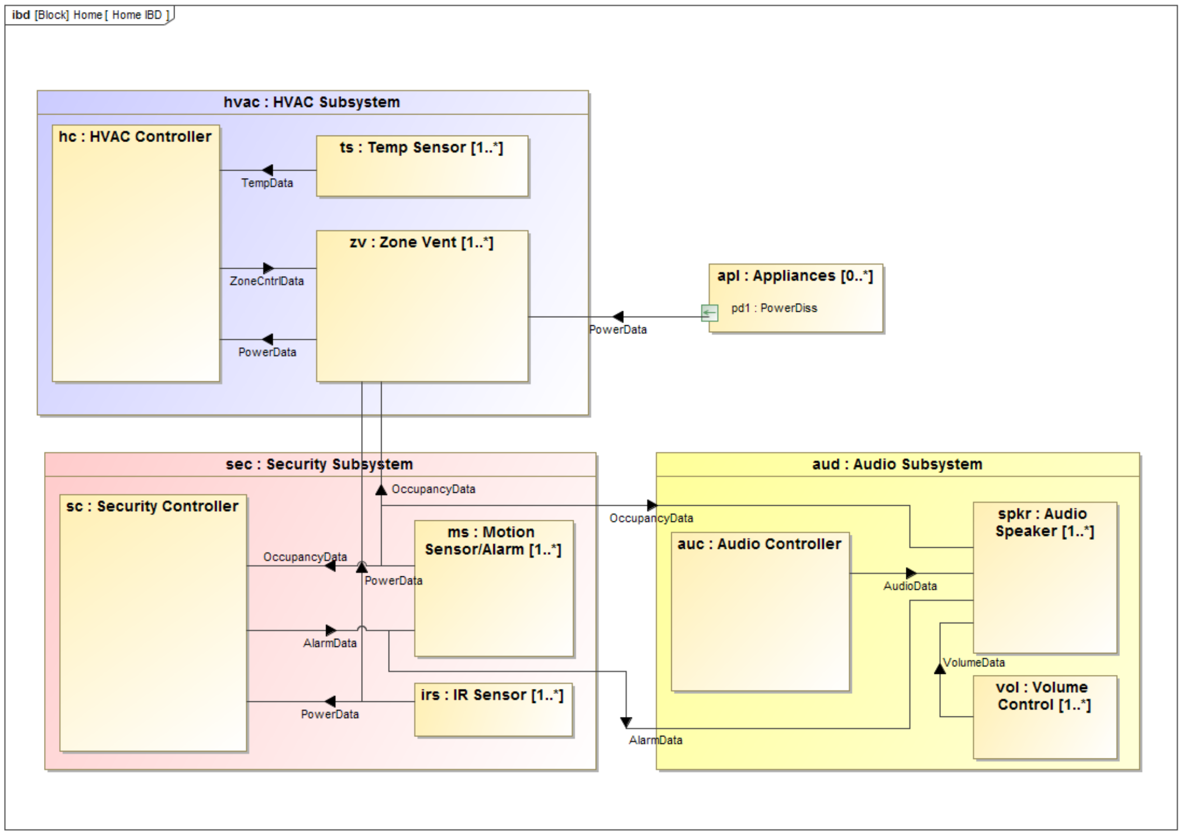


Figure 4 Internal Block Diagram for the Home block

The connectors and item flows shown in Figure 4 must match up with the object flows shown in the Figure 2 activity diagram, making it critical to create and track allocation relationships between them, as shown in Figure 5.

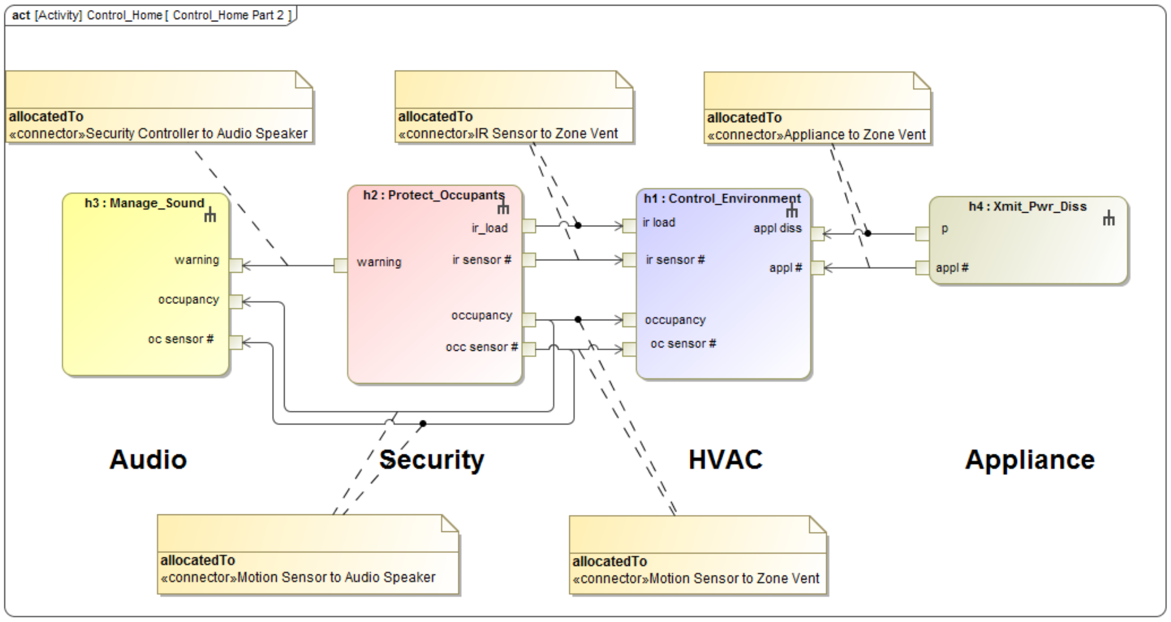
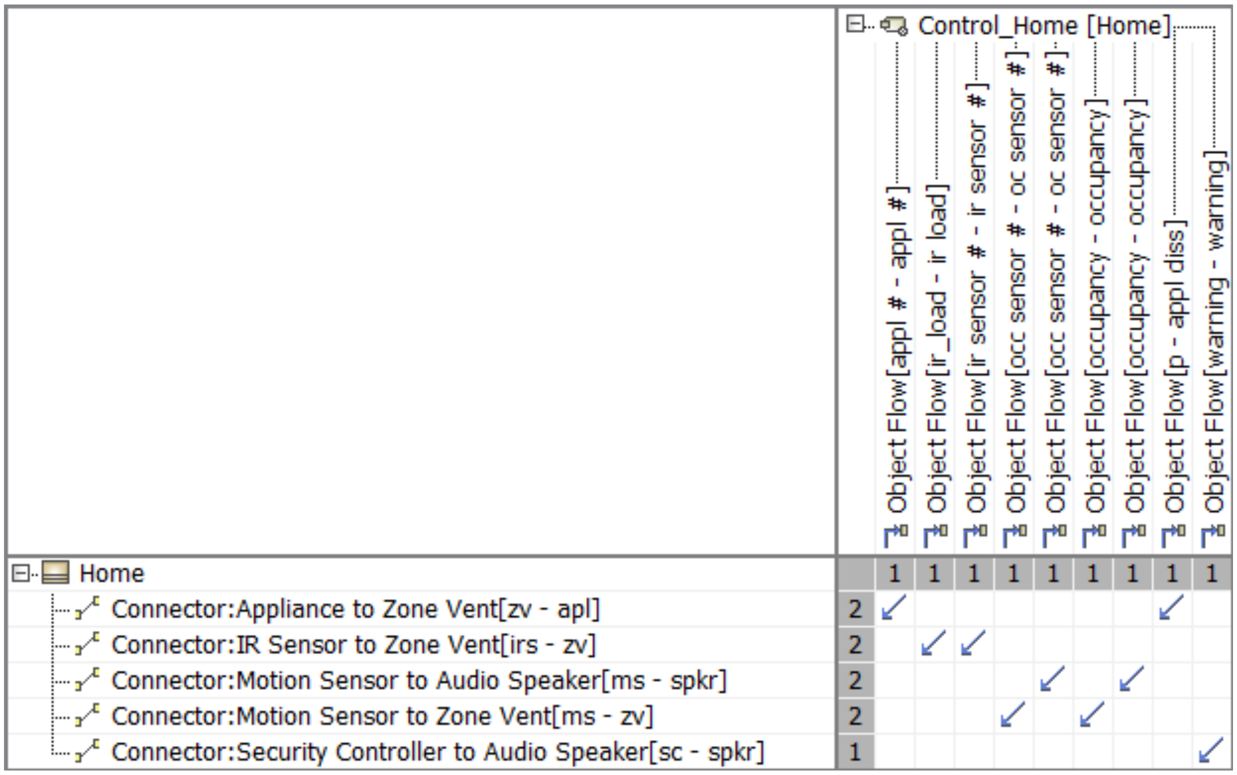


Figure 5 Top-level Activity diagram showing Object Flow Allocations

This tracing can be shown compactly in an Allocation Matrix as in Table 1, which shows the connectors between subsystems as rows and the object flows between corresponding actions as columns. Both sets are fully covered, but note that two object flows, occupancy and occupancy sensor id, are expected to flow over a single connector between motion sensor and zone vent, under an item flow labeled OccupancyData.



		Control_Home [Home]								
		Object Flow[appl # - appl #]	Object Flow[ir_load - ir load]	Object Flow[ir sensor # - ir sensor #]	Object Flow[occ sensor # - oc sensor #]	Object Flow[occ sensor # - oc sensor #]	Object Flow[occupancy - occupancy]	Object Flow[occupancy - occupancy]	Object Flow[p - appl diss]	Object Flow[warning - warning]
Home		1	1	1	1	1	1	1	1	1
Connector:Appliance to Zone Vent[zv - apl]	2	↙							↙	
Connector:IR Sensor to Zone Vent[irs - zv]	2		↙	↙						
Connector:Motion Sensor to Audio Speaker[ms - spkr]	2					↙		↙		
Connector:Motion Sensor to Zone Vent[ms - zv]	2				↙		↙			
Connector:Security Controller to Audio Speaker[sc - spkr]	1									↙

Table 1 Allocation Matrix – Control Home Object Flows to Home Inter-Subsystem Connectors

At this stage, the structural part of the reference architecture has been completed to the extent needed. We have already begun to identify some very specific characteristics of the components needed. For example, the smart zone vent regulates the airflow through it, based on direction from the central HVAC controller, but adjusting locally for both occupancy and heat load as reported by elements of the security subsystem. In ways like this, we can see the potential of the internet-of things being realized.

Before we begin searching for actual components to fulfill these roles, we will consider the questions we want this model to answer and build a parametric model to answer them.

Creating the Parametric Model

For the purposes of this example, we will add two sets of calculations to the model. First, we will calculate the cost of this Smart Home IoT system as the sum of the cost of the system components. Second, we will calculate the power consumption of the HVAC subsystem, taking into account its ability to adjust for occupancy and local heat load.

We will take two different approaches to the parametric modeling. Cost will use an "internal' parametric model, where the mathematical constraints or equations are embedded in the blocks representing the components and subsystems. Power will use an "external" parametric model, where the constraints are contained within specialized analysis blocks which reference the components. Internal models are generally simpler in structure and can be used for roll-ups of intrinsic properties like cost or mass. External models have the advantage that they can be created and modified without changing the basic system structure, which simplifies collaborative modeling. They are useful for specialized analyses using extrinsic properties specific to a particular test case, e.g. the temperature outside the Smart Home.

Cost Model

We want to create a cost model that is independent of the number of rooms or components in the final configuration. We will sum over the one-to-many multiplicities in Figure 1.

Step 1 – We create a new block IotComponent with the value property cost:$. All system components, including the top-level Home block, are made subtypes of this block and inherit this value property (see Figure 6).

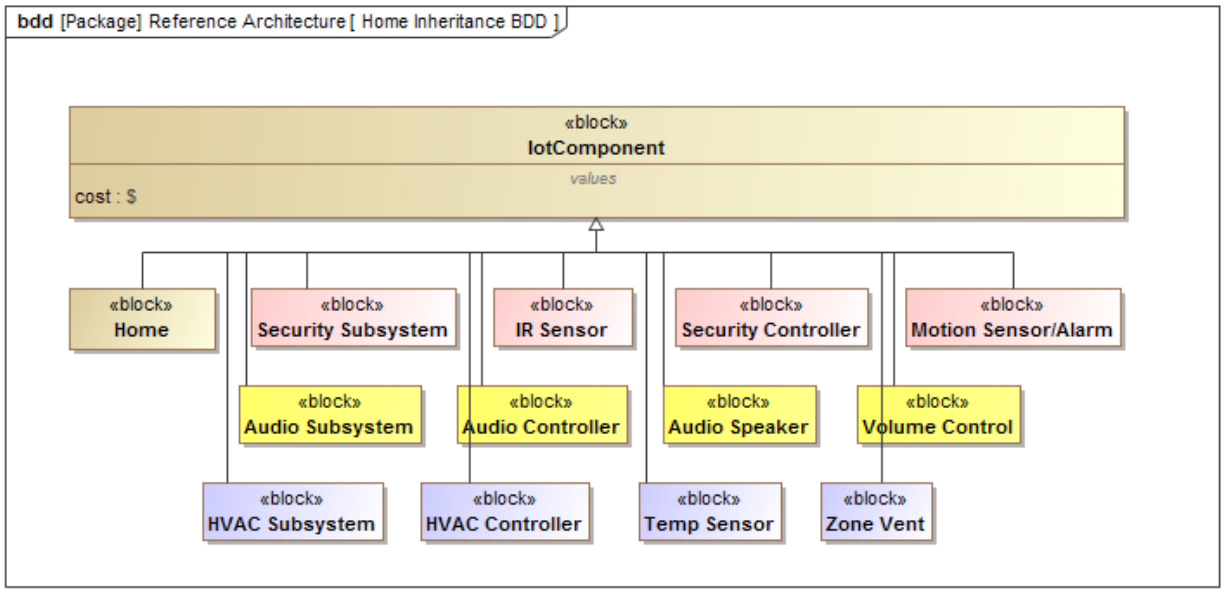


Figure 6 Block Definition Diagram showing inheritance of cost value property by system components

Step 2 – We create a constraint block, CostSum, with the constraint

$$c = c1 + sum(c2) + sum(c3)$$

Step 3 – We create parametric diagrams for the Home block and each of the subsystem blocks, using the CostSum constraint. Figure 7 shows that for the Audio Subsystem block; the other three diagrams are nearly identical. This completes the cost roll-up parametric model.

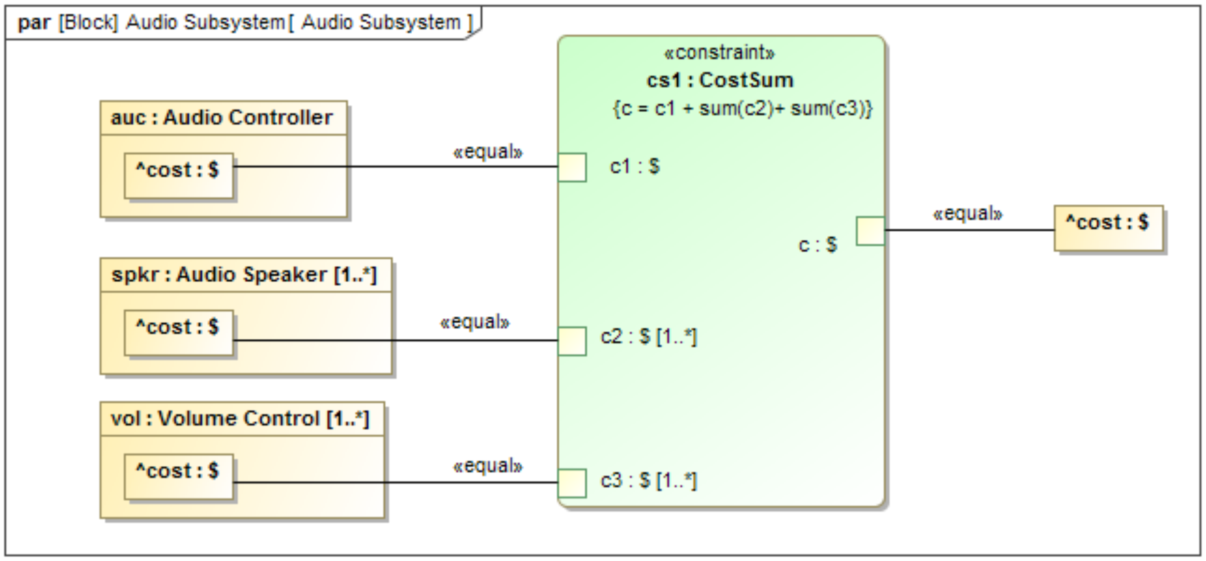


Figure 7 Parametric diagram for Audio Subsystem block

Note: there are more elegant solutions to the problem, given the similarity of the four parametric diagrams. For example, a supertype block containing the parametrics could be created for these four blocks. However, this might limit our flexibility if we substantially changed our structural model, e.g. if one of the subsystems no longer contained three types of parts.

Power Analysis Model

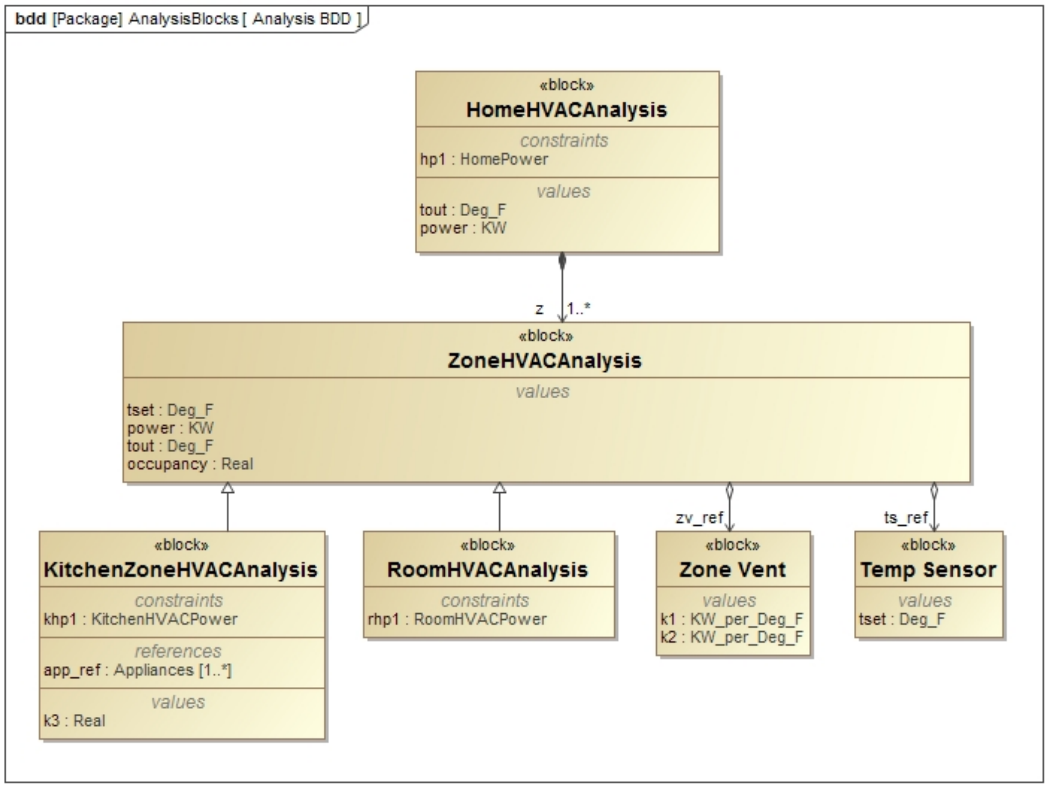


Figure 8 Power Analysis block definition diagram

The Power Analysis model is designed to calculate the power consumed in heating or cooling each room and then sum that value over all rooms. Figure 8 shows the specialized analysis blocks created for this purpose. RoomHVACAnalysis and KitchenZoneHVAC Analysis are subtypes of ZoneHVACAnalysis, which references the Zone Vent and Temperature Sensor assigned to that zone. The kitchen zone also references appliances assigned to such zones, which provide an additional local heat load which must be considered.

Figure 9 shows the parametric model for a basic room. HVAC power consumed is a complex function of the outside temperature and the internal set point of the local thermostat/temperature sensor. k1 and k2 are constants describing the efficiency of heating and cooling, respectively. In reality, these are functions of the size of the room, the placement of the vents, the insulation of the walls and other factors. The values are assigned to the zone vent, because our model doesn't include an element representing the zone/room directly. The power consumption also assumes that power consumption is reduced by an occupancy factor, i.e. the zone isn't heated or cooled when no one is present, as determined by the local motion sensor. While this may not be realistic, a Smart Home may have the capability to determine typically occupancy patterns, using long-term data analyzed in the cloud and sent back to the HVAC Controller.

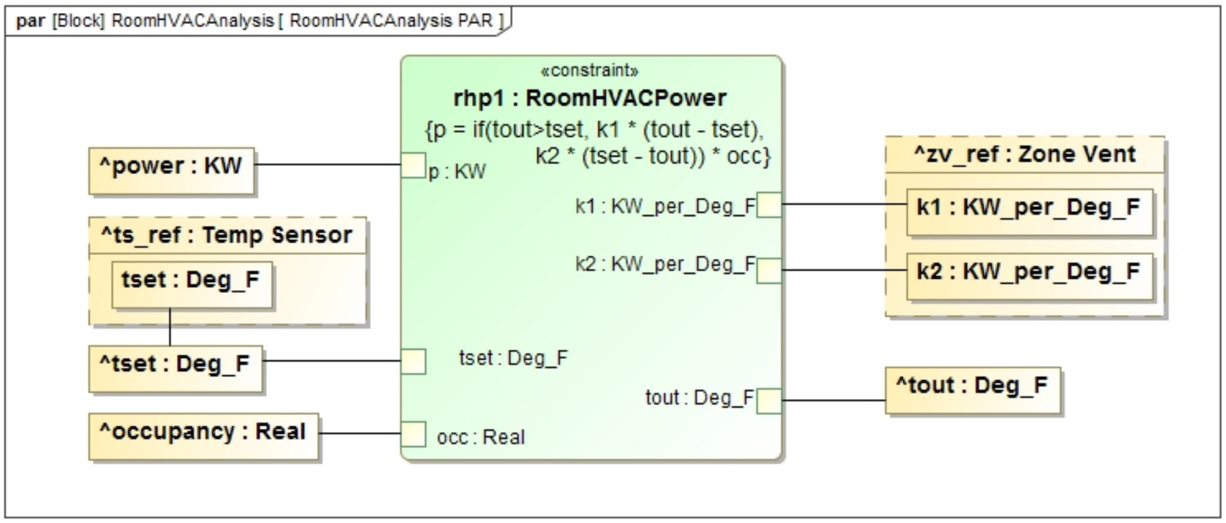


Figure 9 RoomHVACAnalysis Parametric Diagram

The KitchenZoneHVACAnalysis model in Figure 10 is very similar, except that the constraint also includes power dissipation from appliances. These power dissipation values are reported directly from smart appliances to the local zone vent, which incorporates them into its regulation function, i.e. less power needed for heating, more for cooling.

The HomeHVACAnalysis parametric model (not shown) simply sums up the power consumed for HVAC in each of the zones. It also feeds the outside temperature into each of the zone analyses. At some later stage, local power consumption may be part of a larger model to show the effect of smart homes on the local electrical grid.

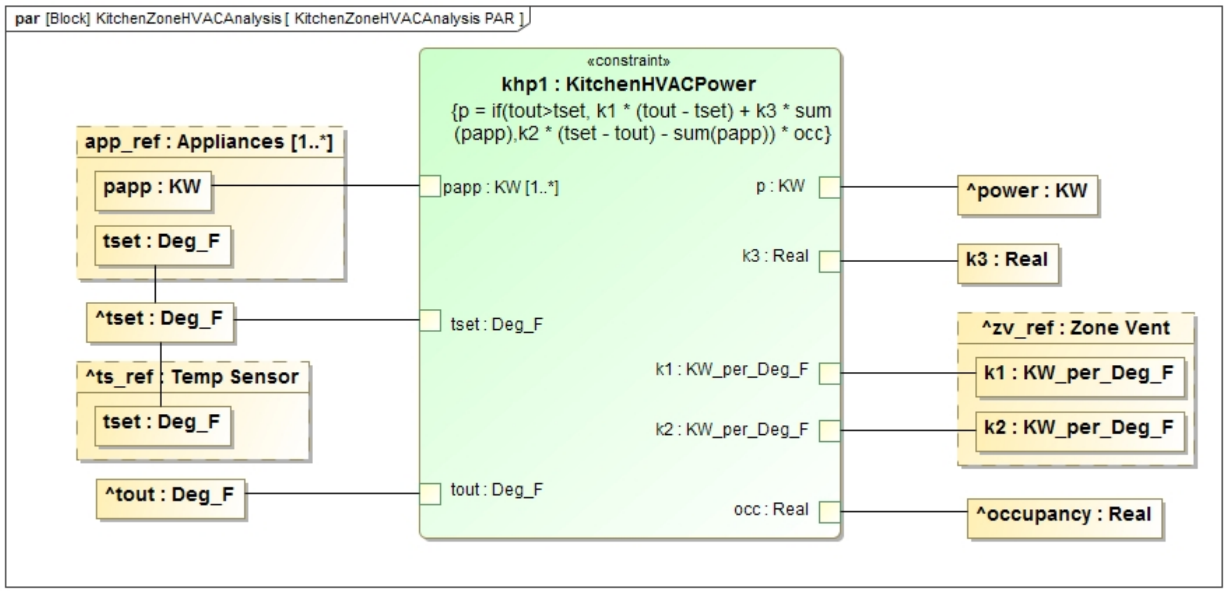


Figure 10 KitchenZoneHVACAnalysis parametric diagram

Connecting to Product Data in External Repositories

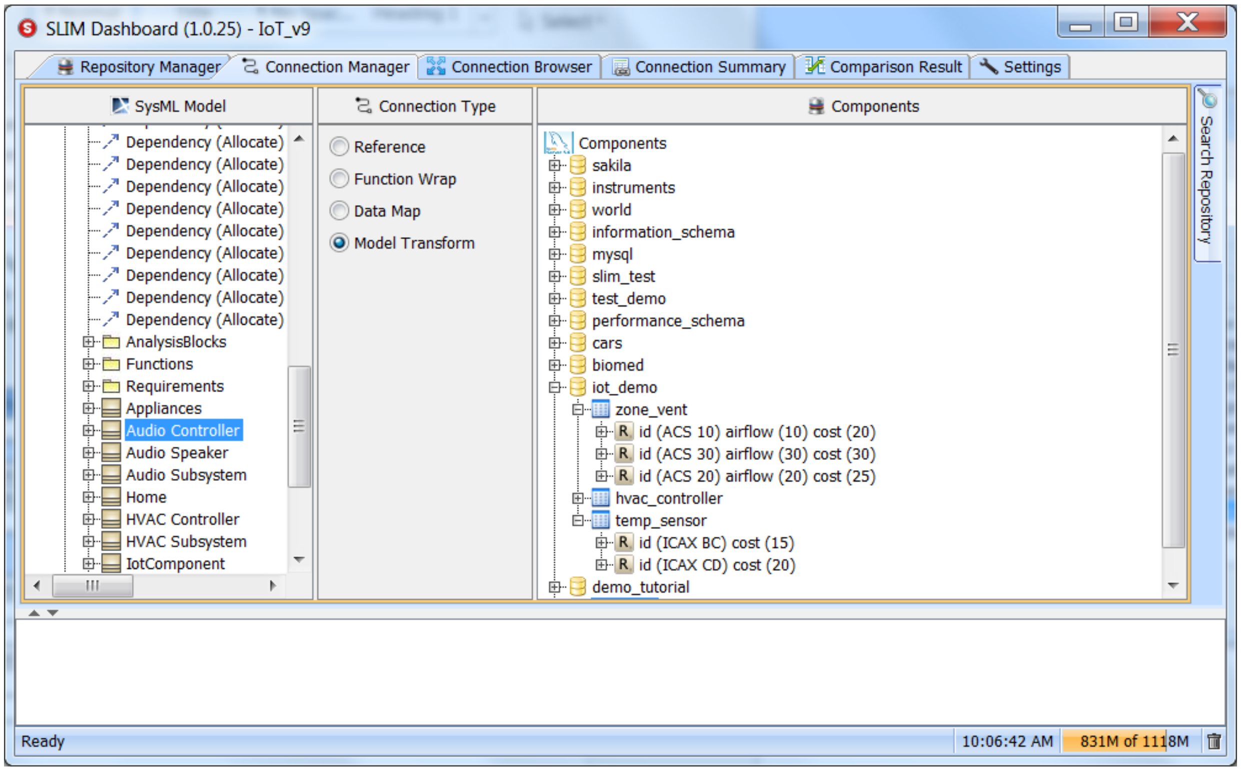
Having completed the reference model, we need to design and evaluate the installation in a specific configuration using real components. At this stage, we take advantage of the availability of software tools that create and maintain linkages between SysML models and external repositories where product data resides. One such tool is Syndeia, an Intercax product that links SysML models in MagicDraw or Rhapsody with a number of commercial databases, PLM systems, CAD and simulation tools.

Table	Name	Value Properties	Proxy Ports	Operations
Appliance				
	Oven	k4(default = 0.01)	pd1:PowerDiss	Xmit_Pwr (p)
	Refrigerator	k4(default = -0.05)	pd1:PowerDiss	xmit_Pwr (p)
Audio Controller				
	Model 100 Audiocon	cost(default = 100)	ap1:AudioCable	create_prgm(out ap1, out sp1)
				encode_prgm(ap1, sp1, out ap2, out sp2)
	Model 150 Audiocon	cost(default = 150)	ap1:AudioProg	create_prgm(out ap1, out sp1)
				encode_prgm(ap1, sp1, out ap2, out sp2)
				xmit_prgm(ap2, sp2, out ap3, out sp3)

Table 2 Product Database (partial)

Table 2 represents part of the contents of a MySQL database containing product data for a variety of IoT components. Each table entry contains the name of the device plus its value properties, its proxy ports and its software operations.

In continuing the IoT system design, we want to be able to pull this product data into the SysML model as specializations of the generic component blocks. These generic SysML blocks (e.g. Audio Controller) each corresponding to a separate table, do not have ports or operations, which are added at the specialization level. They do have value properties corresponding to those in the table, but the default values are redefined at the subtype level. We will see in the next tech note in this series how these specializations will be used to build a specific configuration of the system.


SLIM Dashboard (1.0.25) - IoT_v9

Repository Manager | Connection Manager | Connection Browser | Connection Summary | Comparison Result | Settings

SysML Model
- Dependency (Allocate)
- Dependency (Allocate)
- Dependency (Allocate)
- Dependency (Allocate)
- Dependency (Allocate)
- Dependency (Allocate)
- Dependency (Allocate)
- Dependency (Allocate)
- Dependency (Allocate)
- AnalysisBlocks
- Functions
- Requirements
- Appliances
- Audio Controller
- Audio Speaker
- Audio Subsystem
- Home
- HVAC Controller
- HVAC Subsystem
- IotComponent

Connection Type
- Reference
- Function Wrap
- Data Map
- [✓ Model Transform]

Components
- Components
  - sakila
  - instruments
  - world
  - information_schema
  - mysql
  - slim_test
  - test_demo
  - performance_schema
  - cars
  - biomed
  - iot_demo
    - zone_vent
      - R id (ACS 10) airflow (10) cost (20)
      - R id (ACS 30) airflow (30) cost (30)
      - R id (ACS 20) airflow (20) cost (25)
    - hvac_controller
    - temp_sensor
      - R id (ICAX BC) cost (15)
      - R id (ICAX CD) cost (20)
  - demo_tutorial

Search Repository

Ready | 10:06:42 AM | 831M of 1118M

Figure 11 Syndeia dashboard showing connections to MySQL database with product data

The Syndeia dashboard shown in Figure 11 shows the SysML model in the left column and the MySQL database contents in the right. Individual records in MySQL can be dragged and dropped into the SysML column, bringing the default values of the value properties. Syndeia maintains a persistent connection between the database records and SysML elements, which can be used to update the SysML model as product data in the MySQL database is updated.

Summary

Part 1 of this series explored requirements and functional modeling in SysML for the Internet-of-Things. Part 2 moved on to structural and parametric modeling. A complete reference model of the system has been created and the first step in populating that model with real product information has been carried out. Part 3 will complete the configuration and instantiation of the Smart Home system and use the parametrics to evaluate some key parameters.

About the Author

Dr. Dirk Zwemer (dirk.zwemer@intercax.com) is President of InterCAX LLC, Atlanta, GA and holds OCSMP certification as Model Builder - Advanced.

For further information, visit us at www.intercax.com or contact us at info@intercax.com.