# CPSC 8750

Software Architecture

# SECURITY IN SMART HOME AUTOMATION

Software Architecture Document (SAD)


**CONTENT OWNERS:**

**ASHISH KUMAR SANDIL**

**VIVEK ARAKALI NAGARAJA RAO**

# Table of Contents

# List of Figures



# List of Tables

# 1 Documentation Roadmap

The purpose of this SAD is to define and explore the different architectures and measures taken over the architecture to make the smart home automation model more secure and easy to understand.

## 1.1 Document Management and Configuration Control Information

- Revision Number: << 1 >>
- Revision Release Date: << 4/25/2017 >>
- Purpose of Revision: << N/A>>
- Scope of Revision: <<N/A>>

## 1.2 Purpose and Scope of the SAD

This SAD specifies the software architecture for an artificial pacemaker and its functioning. A clear picture of how the components interact with each other is mentioned in the document with the error ontology**.** The main design architecture used is the feedback control loop which helps clearly describes the working of the pacemaker. All information regarding the software architecture may be found in this document, although much information is incorporated by reference to other documents.

**What is software architecture?** The software architecture for a system[1] is the structure or structures of that system, which comprise software elements, the externally-visible properties of those elements, and the relationships among them [Bass 2003]. "Externally visible" properties refers to those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on. This definition provides the basic litmus test for what information is included in this SAD, and what information is relegated to downstream documentation.

**Elements and relationships**. The software architecture first and foremost embodies information about how the elements relate to each other. This means that architecture specifically omits certain information about elements that does not pertain to their interaction. Thus, a software architecture is an *abstraction* of a system that suppresses details of elements that do not affect how they use, are used by, relate to, or interact with other elements. Elements interact with each other by means of interfaces that partition details about an element into public and private parts. Software architecture is concerned with the public side of this division, and that will be documented

[1] Here, a system may refer to a system of systems.

in this SAD accordingly. On the other hand, private details of elements—details having to do solely with internal implementation—are not architectural and will not be documented in a SAD.

**Multiple structures.** The definition of software architecture makes it clear that systems can and do comprise more than one structure and that no one structure holds the irrefutable claim to being the architecture. The neurologist, the orthopedist, the hematologist, and the dermatologist all take a different perspective on the structure of a human body. Ophthalmologists, cardiologists, and podiatrists concentrate on subsystems. And the kinesiologist and psychiatrist are concerned with different aspects of the entire arrangement's behavior. Although these perspectives are pictured differently and have very different properties, all are inherently related; together they describe the architecture of the human body. So it is with software. Modern systems are more than complex enough to make it difficult to grasp them all at once. Instead, we restrict our attention at any one moment to one (or a small number) of the software system's structures. To communicate meaningfully about an architecture, we must make clear which structure or structures we are discussing at the moment—which *view* we are taking of the architecture. Thus, this SAD follows the principle that documenting a software architecture is a matter of documenting the relevant views and then documenting information that applies to more than one view.

For example, all non-trivial software systems are partitioned into implementation units; these units are given specific responsibilities, and are the basis of work assignments for programming teams. This kind of element will comprise programs and data that software in other implementation units can call or access, and programs and data that are private. In large projects, the elements will almost certainly be subdivided for assignment to sub-teams. This is one kind of structure often used to describe a system. It is a very static structure, in that it focuses on the way the system's functionality is divided up and assigned to implementation teams.

Other structures are much more focused on the way the elements interact with each other at runtime to carry out the system's function. Suppose the system is to be built as a set of parallel processes. The set of processes that will exist at runtime, the programs in the various implementation units described previously that are strung together sequentially to form each process, and the synchronization relations among the processes form another kind of structure often used to describe a system.

None of these structures alone is *the* architecture, although they all convey architectural information. The architecture consists of these structures as well as many others. This example shows that since architecture can comprise more than one kind of structure, there is more than one kind of element (e.g., implementation unit and processes), more than one kind of interaction among elements (e.g., subdivision and synchronization), and even more than one context (e.g., development time versus runtime). By intention, the definition does not specify what the architectural elements and relationships are. Is a software element an object? A process? A library? A database? A commercial product? It can be any of these things and more.

These structures will be represented in the views of the software architecture that are provided in Section **Error! Reference source not found.**.

**Behavior.** Although software architecture tends to focus on structural information, *behavior of each element is part of the software architecture* insofar as that behavior can be observed or discerned from the point of view of another element. This behavior is what allows elements to interact with each other, which is clearly part of the software architecture and will be documented in the SAD as such. Behavior is documented in the element catalog of each view.

## 1.3 Stakeholder Representation

| STAKEHOLDERS | Quality Attribute 1 | Quality Attribute 2 | Quality Attribute 3 |
|---|---|---|---|
| **HOME OWNER** | Safety | Reliability | Availability |
| **MANUFACTURER** | Correctness | Latency | Usability |

## 1.4 Viewpoint Definitions

As described in Section 1.2, a software architecture comprises more than one software structure, each of which provides an engineering handle on different system qualities. A *view* is the specification of one or more of these structures, and documenting a software architecture, then, is a matter of documenting the relevant views and then documenting information that applies to more than one view [Clements 2002].

ANSI/IEEE 1471-2000 provides guidance for choosing the best set of views to document, by bringing stakeholder interests to bear. It prescribes defining a set of viewpoints to satisfy the stakeholder community. A viewpoint identifies the set of concerns to be addressed, and identifies the modeling techniques, evaluation techniques, consistency checking techniques, etc., used by any conforming view. A view, then, is a viewpoint applied to a system. It is a representation of a set of software elements, their properties, and the relationships among them that conform to a defining viewpoint. Together, the chosen set of views show the entire architecture and all of its relevant properties. A SAD contains the viewpoints, relevant views, and information that applies to more than one view to give a holistic description of the system.

The remainder of Section 1.5 defines the viewpoints used in this SAD. The following table summarizes the stakeholders in this project and the viewpoints that have been included to address their concerns.

*Table 1: Stakeholders and Relevant Viewpoints*

| **Stakeholder** | **Viewpoint(s) that apply to that class of stakeholder's concerns** |
|---|---|
| HOME OWNER | Information viewpoint |
| MANUFACTURER | Functional/ Operational/ Development viewpoint |

## 1.4.1 Information Viewpoint Definition

### 1.4.1.1 Abstract

This viewpoint is based on the Home owner's perspectives of how a smart home architecture needs to be designed for security.

### 1.4.1.2 Stakeholders and Their Concerns Addressed

Stakeholders and their concerns addressed include

- The main purpose of having a smart home needs to be served along with having a secure environment
- There is a sense of trust that needs to be maintained as the home owner would have trusted his and the house's content with the manufacturing company.

### 1.4.1.3 Elements, Relations, Properties, and Constraints

The elements in this section include the home controller, client tier, in-range sensor, communicator, appliances. Client tier and Home controller, Home controller and in-range sensor, home controller and communicator, communicator and appliances, in range and appliances are inter related or communicate with each other.

### 1.4.1.4 Language(s) to Model/Represent Conforming Views

Views conforming to the module decomposition viewpoint may be represented by

(a) plain text using indentation or outline form [Clements 2002];

(b) UML, using subsystems or classes to represent elements and "is part of" or nesting to represent the decomposition relation.

#### 1.4.1.5 Applicable Evaluation/Analysis Techniques and Consistency/Completeness Criteria

1) Control-tier is going to send an encrypted info with private key to the home controller. This is message is encrypted and eavesdropping is mitigated

2) Home controller gives to two options either to switch on the sensor or use the user command line to instruct each appliance separately

#### 1.4.1.6 Viewpoint Source

Control Tier and Home Controller

### 1.4.2 Functional Viewpoint Definition

#### 1.4.2.1 Abstract

This viewpoint is based on the Manufacturer's perspectives of how a pacemaker architecture needs to be designed for security.

#### 1.4.2.2 Stakeholders and Their Concerns Addressed

Stakeholders and their concerns addressed include

- Nonfunctioning sensors which fail to detect changes or detect false movements.
- Usage of malicious equipment.

#### 1.4.2.3 Elements, Relations, Properties, and Constraints

The elements in this section include the home controller, client tier, in-range sensor, communicator, appliances. Client tier and Home controller, Home controller and in-range sensor, home controller and communicator, communicator and appliances, in range and appliances are inter related or communicate with each other.

#### 1.4.2.4 Language(s) to Model/Represent Conforming Views

Views conforming to the module decomposition viewpoint may be represented by

(a) plain text using indentation or outline form [Clements 2002];

(b) UML, using subsystems or classes to represent elements and "is part of" or nesting to represent the decomposition relation.

#### 1.4.2.5 Applicable Evaluation/Analysis Techniques and Consistency/Completeness Criteria

a) In range communicates with appliance. In range is a sensor device which on encountering any movement would take the actions described by the user. Ex : Lights
b) Home controller to communicator: user can use different plugins that are authorized by the manufacturer for a secure and integrated communication or use manual commands to operate it. We would highly recommend not to install plugins from unauthorized source.

#### 1.4.2.6 Viewpoint Source

In range sensors and Communicator.

### 1.4.3 Development Viewpoint Definition

#### 1.4.3.1 Abstract

This viewpoint is based on the Developer's perspectives of how a pacemaker architecture needs to be designed for efficiency.

#### 1.4.3.2 Stakeholders and Their Concerns Addressed

Stakeholders and their concerns addressed include

- False interpretations of data
- Packets lost during communication

#### 1.4.3.3 Elements, Relations, Properties, and Constraints

The elements in this section include the home controller, client tier, in-range sensor, communicator, appliances. Client tier and Home controller, Home controller and in-range sensor, home controller and communicator, communicator and appliances, in range and appliances are inter related or communicate with each other.

#### 1.4.3.4 Language(s) to Model/Represent Conforming Views

Views conforming to the module decomposition viewpoint may be represented by

(a) plain text using indentation or outline form [Clements 2002];

(b) UML, using subsystems or classes to represent elements and "is part of" or nesting to represent the decomposition relation.

#### 1.4.3.5 Applicable Evaluation/Analysis Techniques and Consistency/Completeness Criteria

1) Communicator communicates with the appliances. The request sent from the plugins can be easily directed towards the appliance to be executed

2) For manual commands, communicator needs to interpret the commands and then change the status of the appliance accordingly.

#### 1.4.3.6 Viewpoint Source

Communicator

### 1.4.4 Operational Viewpoint Definition

#### 1.4.4.1 Abstract

This viewpoint is based on the Customer's perspectives of how a pacemaker architecture needs to be designed for efficiency.

#### 1.4.4.2 Stakeholders and Their Concerns Addressed

- Unauthorized access requests

#### 1.4.4.3 Elements, Relations, Properties, and Constraints

The elements in this section include the home controller, client tier, in-range sensor, communicator, appliances. Client tier and Home controller, Home controller and in-range sensor, home controller and communicator, communicator and appliances, in range and appliances are inter related or communicate with each other.

#### 1.4.4.4 Language(s) to Model/Represent Conforming Views

Views conforming to the module decomposition viewpoint may be represented by

(a) plain text using indentation or outline form [Clements 2002];

(b) UML, using subsystems or classes to represent elements and "is part of" or nesting to represent the decomposition relation.

#### 1.4.4.5 Applicable Evaluation/Analysis Techniques and Consistency/Completeness Criteria

a) Appliances communicate either the client tier or in range sensors, which is strictly a one-way communication and notifies any change in the state of the device. This helps use to identify any unauthorized requests in the smart home system.

#### 1.4.4.6 Viewpoint Source

Client tier

## 1.5 How a View is Documented

- Section 3.1: Information View.
- Section 3.1.1: View description. This view is in direct relation to a home owner stakeholder. This brings out the information needed for an optimized architecture.
- Section 3.1.2: View packet overview. The product is useless to the company and the consumer if it does not do what is expected.
- Section 3.1.3: Architecture background. Feedback control design pattern
- Section 3.1.4: Variability mechanisms. No variations or constraints

- Section 3.2: Functional View.
- Section 3.2.1: View description. This view is in direct relation to a manufacturer stakeholder. This brings out the information needed for an optimized architecture.
- Section 3.2.2: View packet overview. Proper functioning of the system
- Section 3.2.3: Architecture background. Feedback control design pattern
- Section 3.2.4: Variability mechanisms. No variations or constraints

- Section 3.3: Development View.
- Section 3.3.1: View description. This view is in direct relation to a home owner stakeholder. This brings out the information needed for an optimized architecture.
- Section 3.3.2: View packet overview. The packet gives a holistic view
- Section 3.3.3: Architecture background. Feedback control design pattern
- Section 3.3.4: Variability mechanisms. No variations or constraints

- Section 3.4: Operational View.
- Section 3.4.1: View description. This view is in direct relation to a manufacturer stakeholder. This brings out the information needed for an optimized architecture.

- Section 3.4.2: View packet overview. The final product and its pros and cons.
- Section 3.4.3: Architecture background. Feedback control design pattern
- Section 3.4.4: Variability mechanisms. No variations or constraints

## 1.6 Relationship to Other SADs

Not applicable

## 1.7 Process for Updating this SAD

Once the issue has been sent it will be scrutinized within a week and an E-mail will be sent confirming the version and update.

Kindly use the below format to send the E-mail.

Name:

Version being reported:

Date:

Email Address:

Brief Description of discrepancies, errors, inconsistencies or omissions:

Email Address:

To be sent to: varakal@g.clemson.edu or asandil@g.clemson.edu

Subject line: SAD Reporting

# 2 Architecture Background

## 2.1 Problem Background

- The Samsung smart home, Prosyst, YunFei Robotics Laboratory, WSO2, e-Zest are some of the well-known architectures for a smart home. But they are prone to malicious attacks from hackers which make it difficult for home owners as they are prone to data and resource loss.
- A security in a smart home is the main goal of a systematic architecture.
- The defected flows or errors can be determined by building the architecture first and then implementing it.

### 2.1.1 System Overview

The smart home architecture consists of five basic components which interact with each other. The components include home controller, client tier, in-range sensor, communicator, appliances

### 2.1.2 Goals and Context

- Building an architecture which reduces complexity and increases efficiency.
- Adding or removing components to check security of the smart home.
- Reducing malfunctioning by determining risks and errors.

### 2.1.3 Significant Driving Requirements

The quality attributes which were selected during a QAW are as follows:

Home Owner: Safety, Reliability, Availability

Manufacturer: Correctability, latency, usability

## 2.2 Solution Background

### 2.2.1 Architectural Approaches

- Use the design pattern feedback control loop.

- Add plugin and event manager threads to help improve the security.
- Give a holistic and detailed approach at the same time using different implementations to help both a lay man and an expert evaluate the architecture.

### 2.2.2 Analysis Results

Compared to other design patterns the feedback control loop models the smart home in a way that it is easily understandable and efficient to construct.

### 2.2.3 Requirements Coverage

Home Controller:

### 2.2.4 Summary of Background Changes Reflected in Current Version

Through this architecture we have minimized the risk which were caused due to eavesdropping attack, Mirai attack, Zigbee attack, Z Wave attack, KNX attack and hardware vulnerability attack.

## 2.3 Product Line Reuse Considerations

Any further revisions can be carried out on the current decision architecture.

- The feedback control loop is open to additions and removals.
- The components which have been added are specific and carry out the operations efficiently without wastage of resources.
- DOS and DDOS attacks need to be addressed in the next coverage

# 3 Referenced Materials


| | |
|---|---|
| Barbacci 2003 | Barbacci, M.; Ellison, R.; Lattanze, A.; Stafford, J.; Weinstock, C.; & Wood, W. *Quality Attribute Workshops (QAWs)*, Third Edition (CMU/SEI-2003-TR-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr016.html>. |
| Bass 2003 | Bass, Clements, Kazman, *Software Architecture in Practice*, second edition, Addison Wesley Longman, 2003. |
| Clements 2001 | Clements, Kazman, Klein, *Evaluating Software Architectures: Methods and Case Studies*, Addison Wesley Longman, 2001. |
| Clements 2002 | Clements, Bachmann, Bass, Garlan, Ivers, Little, Nord, Stafford, *Documenting Software Architectures: Views and Beyond*, Addison Wesley Longman, 2002. |
| IEEE 1471 | ANSI/IEEE-1471-2000, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, 21 September 2000. |

# 4 Directory

## 4.1 Glossary

| Term | Definition |
|---|---|
| software architecture | The structure or structures of that system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [Bass 2003]. "Externally visible" properties refer to those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on. |
| view | A representation of a whole system from the perspective of a related set of concerns [IEEE 1471]. A representation of a particular type of software architectural elements that occur in a system, their properties, and the relations among them. A view conforms to a defining viewpoint. |
| view packet | The smallest package of architectural documentation that could usefully be given to a stakeholder. The documentation of a view is composed of one or more view packets. |
| viewpoint | A specification of the conventions for constructing and using a view; a pattern or template from which to develop individual views by establishing the purposes and audience for a view, and the techniques for its creation and analysis [IEEE 1471]. Identifies the set of concerns to be addressed, and identifies the modeling techniques, evaluation techniques, consistency checking techniques, etc., used by any conforming view. |

## 4.2 Acronym List

| | |
|---|---|
| API | Application Programming Interface; Application Program Interface; Application Programmer Interface |
| ATAM | Architecture Tradeoff Analysis Method |
| CMM | Capability Maturity Model |
| CMMI | Capability Maturity Model Integration |
| CORBA | Common object request broker architecture |
| COTS | Commercial-Off-The-Shelf |
| EPIC | Evolutionary Process for Integrating COTS-Based Systems |
| IEEE | Institute of Electrical and Electronics Engineers |
| KPA | Key Process Area |
| OO | Object Oriented |
| ORB | Object Request Broker |
| OS | Operating System |
| QAW | Quality Attribute Workshop |
| RUP | Rational Unified Process |
| SAD | Software Architecture Document |
| SDE | Software Development Environment |
| SEE | Software Engineering Environment |
| SEI | Software Engineering Institute<br>Systems Engineering & Integration<br>Software End Item |
| SEPG | Software Engineering Process Group |
| SLOC | Source Lines of Code |
| SW-CMM | Capability Maturity Model for Software |
| CMMI-SW | Capability Maturity Model Integrated - includes Software Engineering |
| UML | Unified Modeling Language |

# 5 Sample Figures & Tables

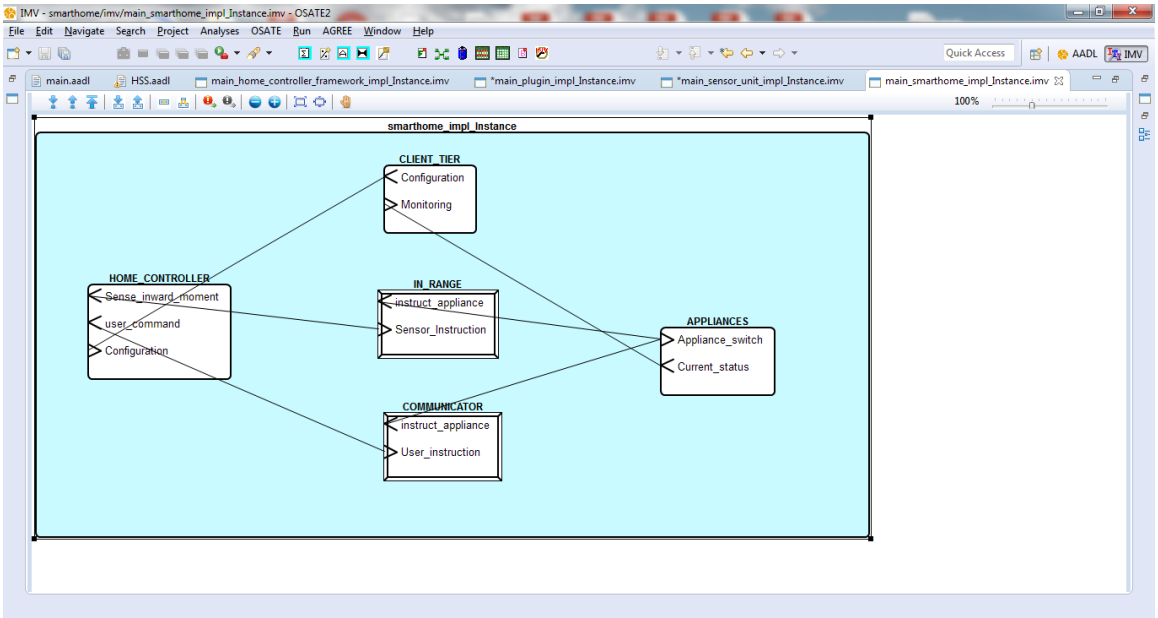


Figure 1: Feedback- Control Loop

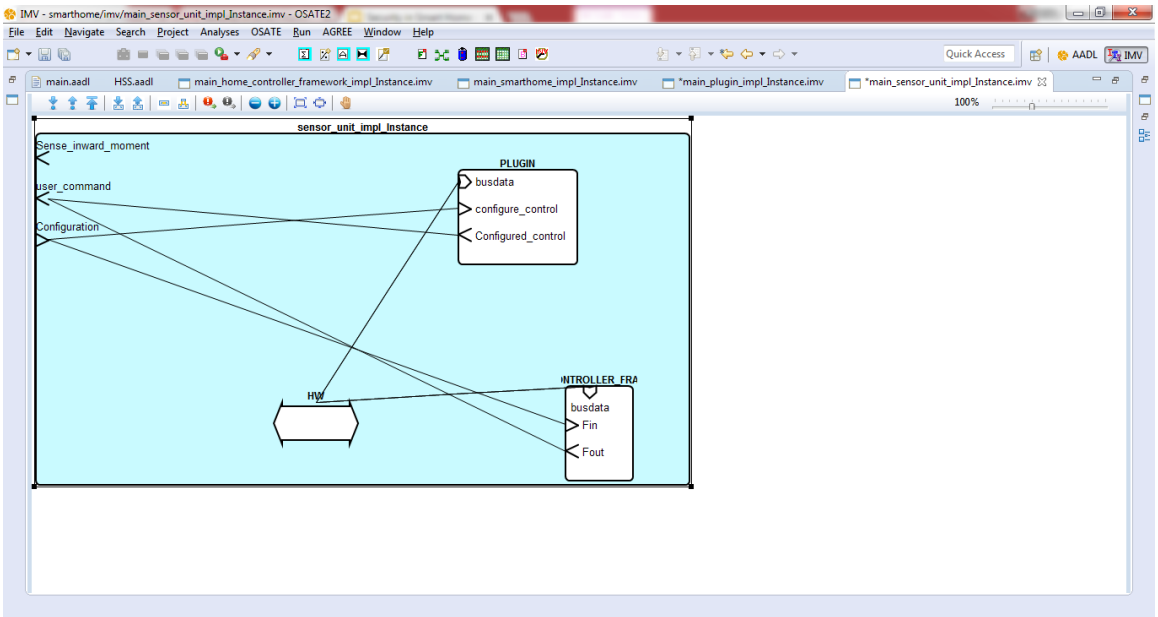


Figure 2: Graphical Representation of Sensor Unit using the AADL model

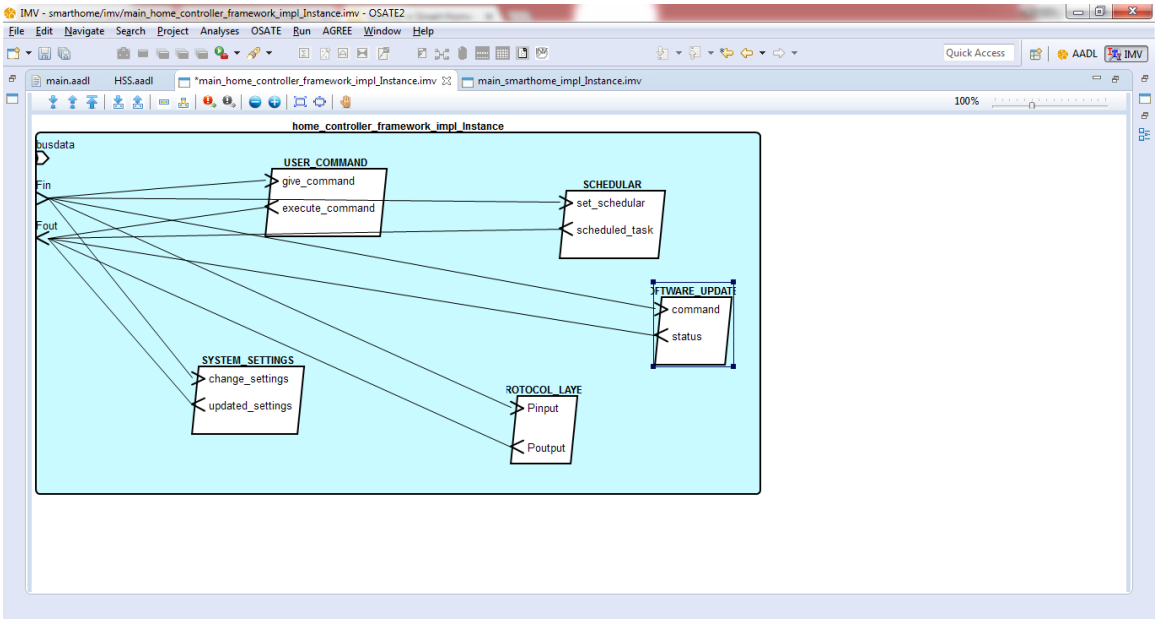


Figure 3: Graphical Representation of Home Controller using the AADL model

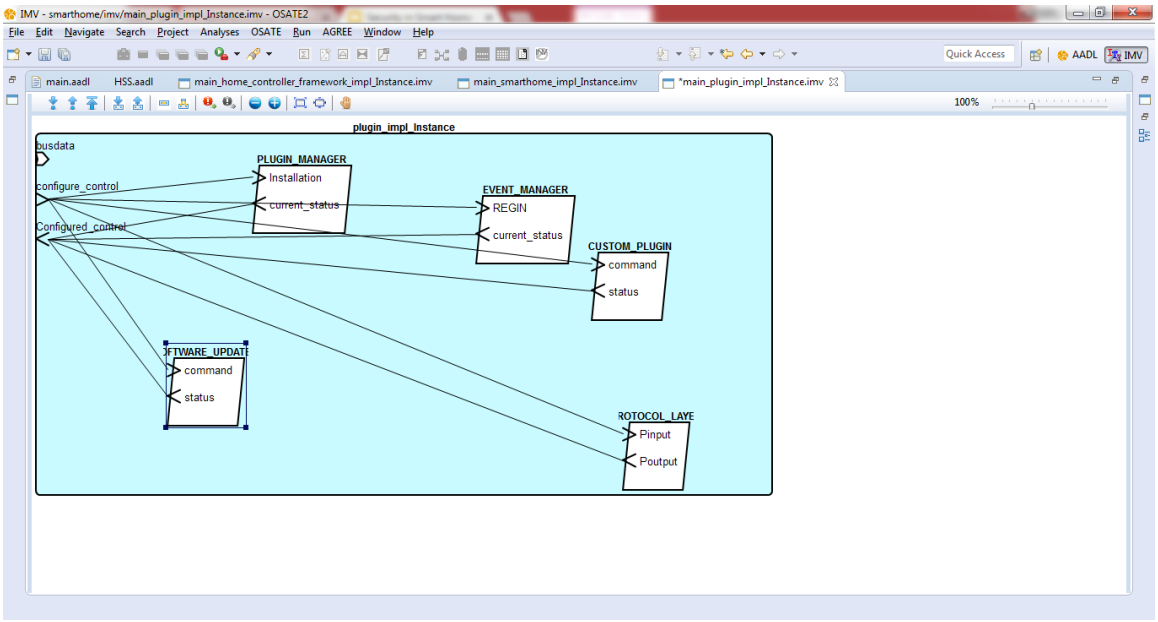


Figure 4: Graphical Representation of Plugins using the AADL model

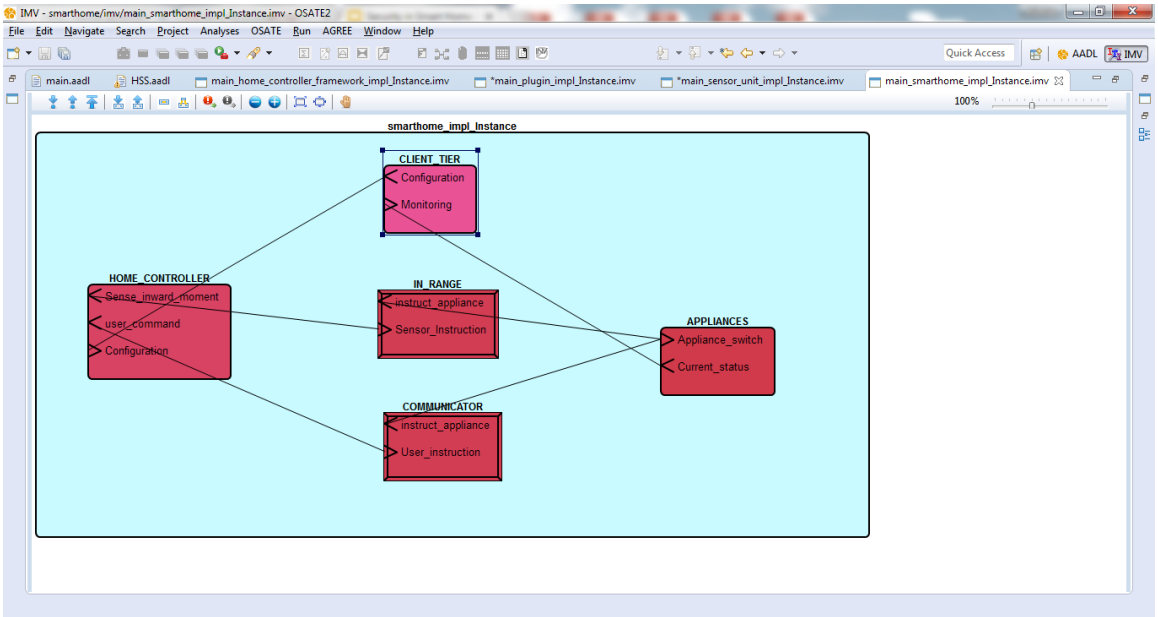


Figure 5: Error Model

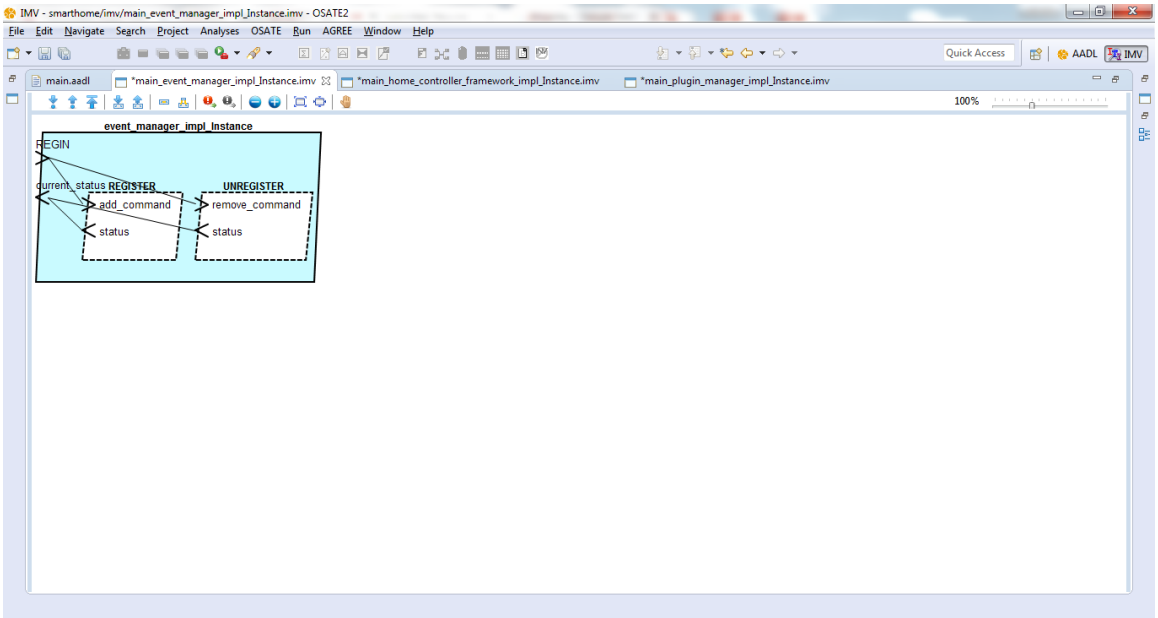


Figure 6: Event Manager Thread

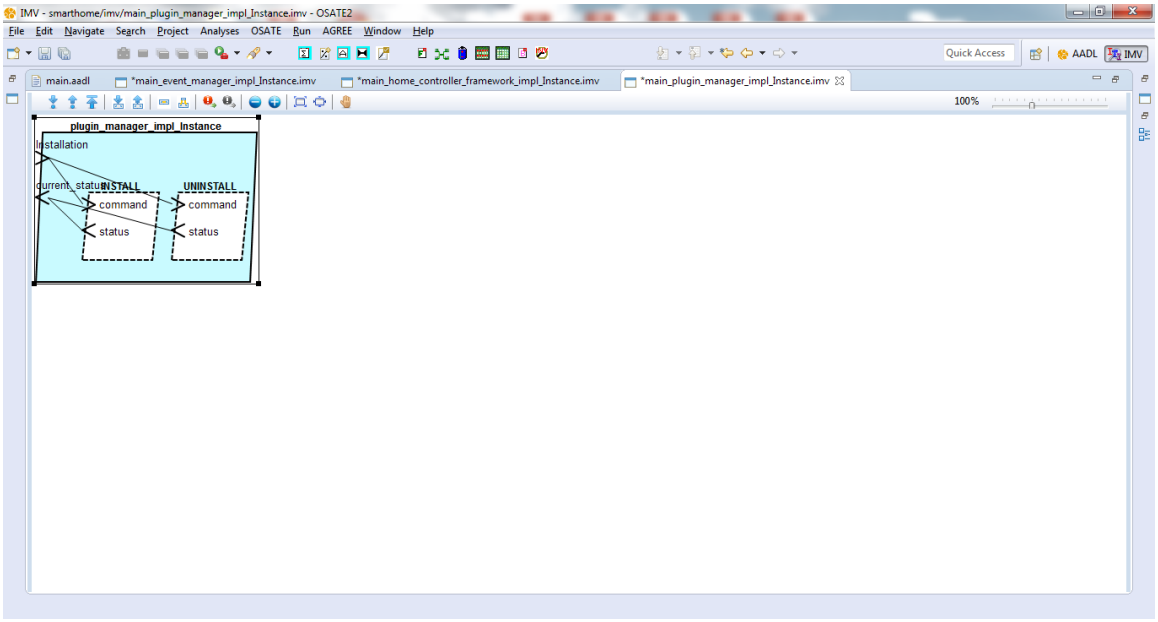


Figure 7: Plugin Manager Thread

| Quality Attributes | HOME OWNER | MANUFACTURER | All | Ranking |
|---|---|---|---|---|
| efficiency | | | | |
| time economy independence | | | | |
| resource economy independence | | | | |
| functionality | 1 | 1 | 2 | |
| completeness | | | | |
| correctness | | | | |
| security / safety | 8 | 4 | 12 | 1 |
| compatibility | | | | |
| maintainabilty | | | | |
| interoperability | | | | |
| correctability | 3 | 4 | 7 | 2 |
| understadnability | | | | |
| expandability | | | | |
| testability | | | | |
| usability | | 3 | 3 | 6 |
| installability | | | | |
| reusability | | | | |
| reliability | 4 | 1 | 5 | 3 |
| non-deficiency | | | | |
| error tolerance | - | 1 | 1 | |
| availability | 4 | | 4 | 5 |
| latency | | 5 | 5 | 4 |
| ease of learning | | 1 | 1 | - |
| operabilty | | | | - |
| communicativeness | | | | - |
| **Total Votes:** | 20 | 20 | 40 | |
| **Remaining Votes:** | 0 | 0 | 0 | |

Table 1: QAW voting table