Title:	*D8.5.2 - Modelling Smart City Urban Safety Planner - Final prototype design*
Authors:	*Florin Picioroaga (Siemens), Septimiu Nechifor (Siemens)*
Editor:	*Florin Picioroaga (Siemens)*
Reviewers:	*Elisabetta Di Nitto (Polimi), Peter Matthews (CA)*
Identifier:	*Deliverable # D8.5.2*
Nature:	*Report*
Version:	*1.0*
Date:	*15/10/2014*
Status:	*Final*
Diss. level:	*Public*

Executive Summary

This deliverable presents the final design of the Smart City Urban Safety Planner. The initial design of selected case study has been updated in order to reflect the prototyped features of the application as it is depicted in delivery D9.4.1 and upcoming D9.4.2. An updated architecture is presented, ready to support Storm cluster deployment on multi-cloud environments.
Regarding the tools developed inside the project, MODACloudsIDE has been used to sketch the architecture and to generate the deployment models while CloudML to validate the resulted models. This document presents the updated design model of our case study also in an evolved version of CloudML supporting specific features.
The document describes also the planned use and demonstration for the Monitoring components and Hegira4Cloud to be executed between M25 and M30.

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 318484 (MODAClouds).

Members of the MODAClouds consortium:

Politecnico di Milano	Italy
StiftelsenSintef	Norway
InstitutulE-Austria Timisoara	Romania
Imperial Collegeof Science, Technologyand Medicine	United Kingdom
SOFTEAM	France
Siemens srl	Romania
BOC Information Systems GMBH	Austria
Flexiant Limited	United Kingdom
ATOS Spain S.A.	Spain
CA Technologies Development Spain S.A.	Spain

Published MODAClouds documents

These documents are all available from the project website located at http://www.modaclouds.eu/

Contents

Chapter 1

Introduction

1.1 Context and objectives

This document, referred as D8.5.2, describes the final design of the Smart City Urban Safety Planner (SCUSP) case study.
Aspects presented bellow represents an evolution of architecture description for SCUSP case study. Based on SCUS identified needs the architecture consider project tools that were released at M12 and their evolved version delivered at M18. Also, the architectural design was updated based on the achievements of the delivery D9.4.1 having implementation effort consumed and experiences as a "reality check".
This document is relevant for the ongoing development phase as it will serve as the basis for the final implementation of the prototype that will be delivered at the M36.
In order to leverage the case study complexity and to fit in the project plan, as difference from the previous version D8.5.1, some key architectural decisions were taken such as:

- Design with MODACloudsIDE only the artefacts that were presented in the delivery D9.4.1 (namely: *GasDataAnalyzer*, *SCUSPAdmin* and *Reconcliator*)
- Use Cassandra as Database as Service (DBaaS) for the data storage of the application
- Plan the usage of Monitoring tool capabilities

For development purpose, a single Cassandra node will be installed manually on Amazon EC2.

1.2 Structure of the document

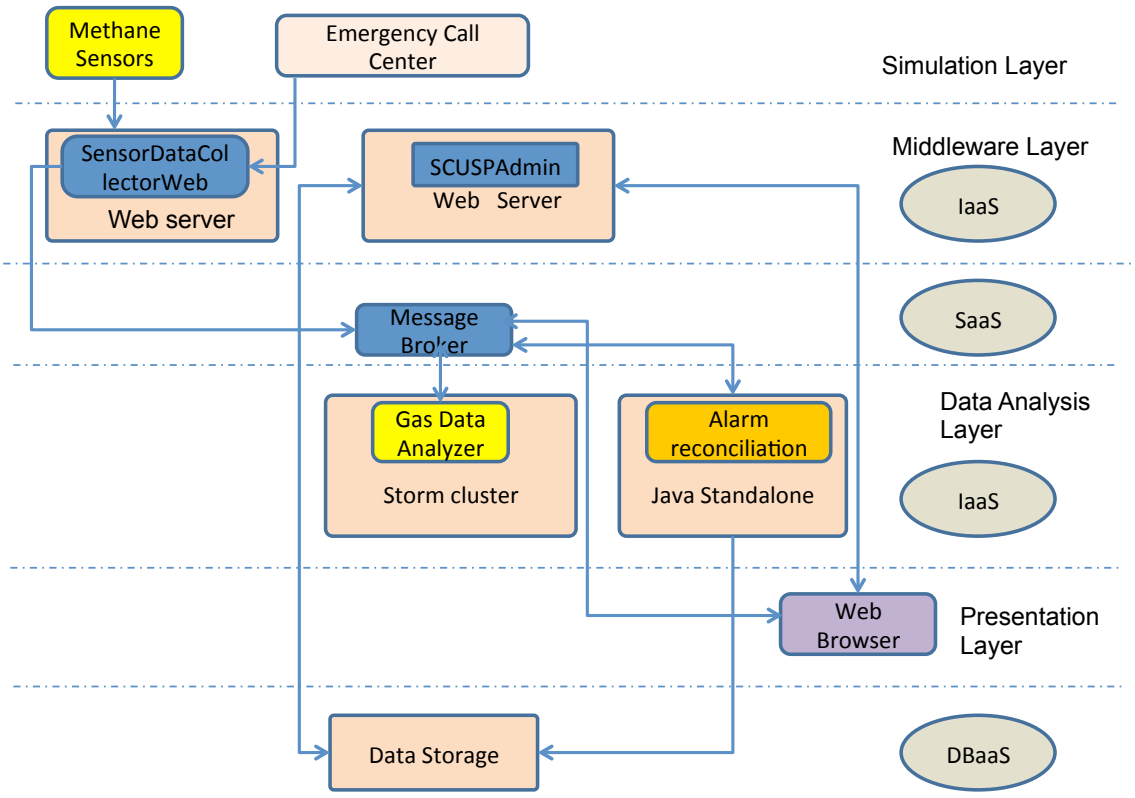
The document is structured as follows:

- Chapter 2 gives an overview of architectural concept for the SCUSP application
- Chapter 3 describes the role of the MODAClouds tools for designing the case study
- Chapter 4 concludes the document
- The Appendix provides detailed design of our application and also the used data models

Chapter 2

Design of Smart City Urban Planner

In order to leverage the complexity of the architecture depicted the previous delivery D8.5.1, a simplified version based on D9.4.1 will be the base of our implementation. Due to the fact that access to real time real data from a gas utility provider is sensitive from security, privacy and commercial perspective and also from the emergency center role one, in order to prove our prototype architecture we have used a simulated data generator based on public domain statistics in order to model and generate data with a good accuracy level. Accuracy is relevant for both velocity and volume of data for the types of premises considered. This is the role played by the *Simulation Layer* that sends data via a web service downstream to the analyzers components. The workflows between the involved components are detailed in the D8.5.1. Bellow picture depicts the final architecture version of our case study.



As it can be observed, the entire communication between the distributed components is done via *MessageBroker*. In the context of this prototype we will use a dedicated installation of the *RabbitMQ* on the cloud provider virtual machine, but in the production it is intended to use *CloudAMQP* cloud service.

Chapter 3

Achievements

Objectives	Achievements
Update the SCUSP architecture and deployment models to support multi-cloud and Storm cluster distributed infrastructure	Section 3.1.1 describes how we have updated our architecture design at the CPIM and CPSM level in the MODACloudsIDE in order to deploy a Storm cluster on multiple clouds. The Appendix provides the up-to-date architectural design aspects of our case study.
Define the scenario for the Hegira4Cloud NOSQL migration tool	Section 3.1.2 presents the usage of the Hegira4Cloud in the context of migration of a dataset containing historical traffic data.
Provide the data model for SCUSP and define the format of the telegrams	In the Appendix, we provide the draft data models for: • gas pipes isolation scenario • traffic data • the data sent by the gas sensors and emergency center (telegram)
Define the role of the monitoring tools use	Section 3.1.2 presents the plans to use the monitoring infrastructure and target features considered.

3.1 Role of MODAClouds in the Case Study

Due to the start of both design and implementation of use case the architectural thinking of SCUSP reached a level of maturity due to a better understanding of technological options and associated benefits. Therefore, we made some options that, in our opinion, bring value and flexibility. Same time, selected options – like usage of Storm cluster on multiple cloud contexts, raised interesting research challenges toward considered MODAClouds tools.
SCUSP is use promote a step wise approach where the feedback towards tools work packages is provided one by one, leading to a solid list of issues solved and enhances delivered in the tools.
A number of tools have been already considered (CloudML and MODAClouds IDE) and are plans for other ones (Monitoring and Hegira4Cloud) in the very near future.
As reference deployment platforms, SCUSP team selected Amazon EC2 and Flexiant.

3.1.1 Usage of MODAClouds IDE tool

For the initial prototype of the SCUSP application, described in the delivery D9.4.1, we have used CloudML solely without any tool for creating the deployment model. We have taken this approach in order to be sure that CloudML is able to deploy our application on the cloud. This approach helped us to find earlier the bugs and issues associated with this tool. Our application, being a high distributed system with intricate relationships between the components, required a lot of effort to create and to validate the CloudML model. Moreover, manually editing the model and understanding the relationships between its artefacts, involved the inherent drawbacks from client perspective such as typos and wrong usage of the concepts. Therefore, a tool for avoiding such kind of problems was necessary. MODACloudsIDE provides such a feature to design and to generate this model for us.

As far as we have proven that our model is stable and able to produce the expected result (meaning the deployment on the specified cloud providers), then MODAClouds IDE came handy for us.
In order to achieve our goal for creating and generating the deployment model, with the help of our partners, Softeam, we have redesigned our application artefacts at the CPIM and CPSM level.
Based on D9.4.1, the main artefacts of the architecture are:

- smm1 – node containing the Nimbus Storm master and Zookeeper master
- sww1 – node depicting the Storm worker
- TomcatScusp9 – node containing the *SCUSPAdmin* web application
- Recon1 – node containing the *Reconciliator* application

Those artefacts are depicted bellow in the associated CPIM model generated using MODAClouds IDE reflecting also considered installation dependencies between different technological components used and their associated distribution along virtual machines involved.

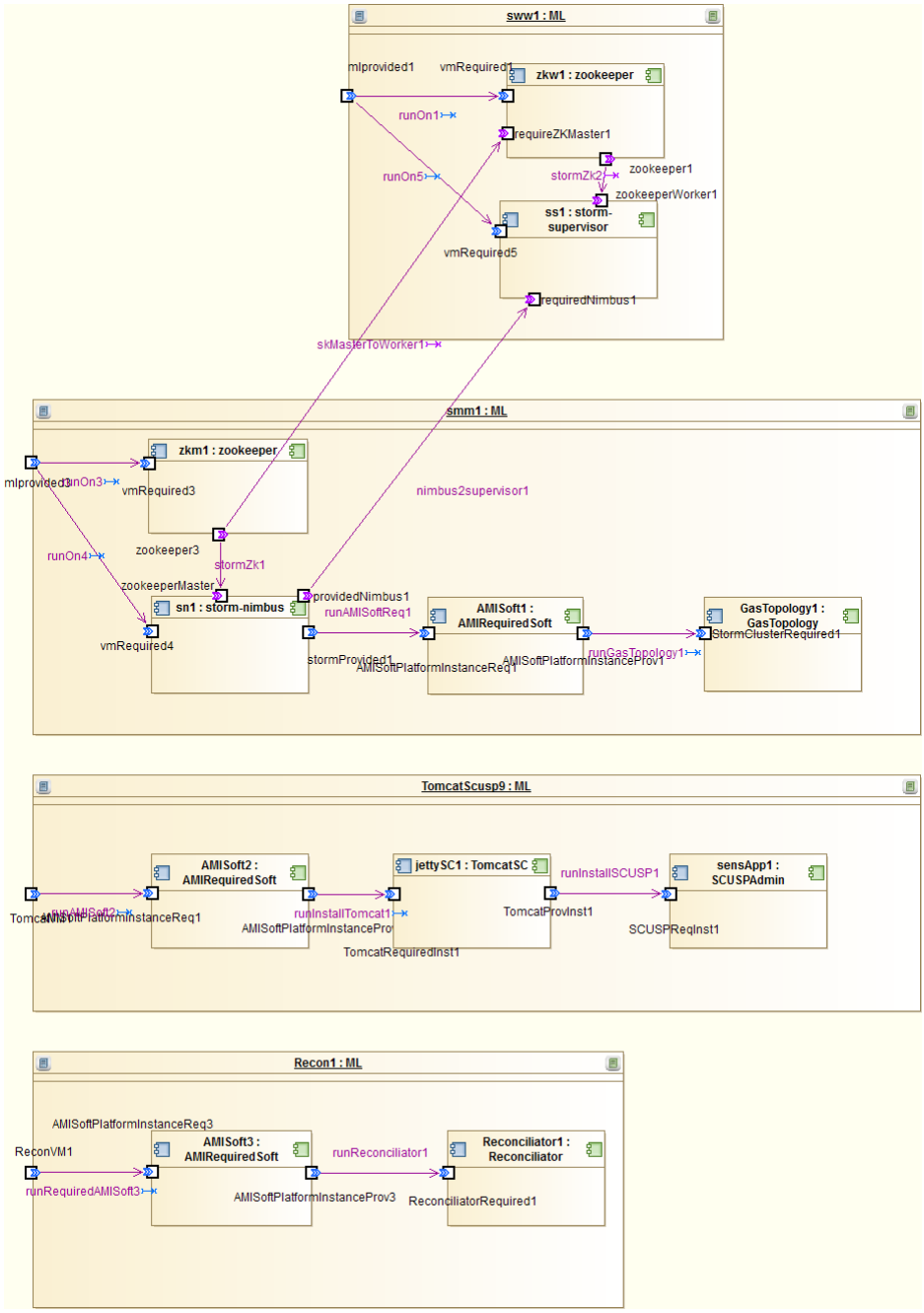


Figure 2 SCUSP' CPIM architecture

As stated before current architecture considers also the initial results in design and implementation phase. As a novelty from the delivery D8.5.1, we plan to deploy the Storm part (stream processing) of our application on Flexiscale not only on Amazon. With the help of MODAClouds IDE we had to add different cloud providers to our CPSM model and to add a supported VM from this provider.

Thanks to the multi-cloud deployment we will increase the availability of the most critical part of our application, namely the streaming processing engine. In the appendix the reader will find the deployment model for Flexiscale platform.

Bellow is presented the deployment diagram of the SCUSP application, and it's details are detailed in D9.4.1.

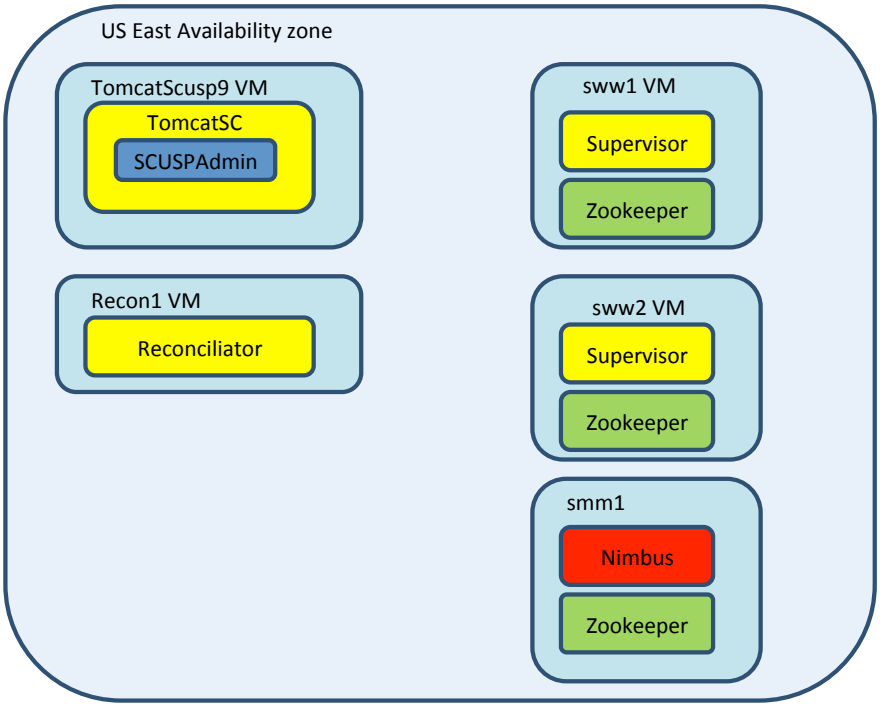


Figure 3: Deployment overview of the SCUSP on Amazon EC2

Related to this conceptual diagram the following CPSM level image was generated via Modelio regarding deployment model for Amazon EC2 case.

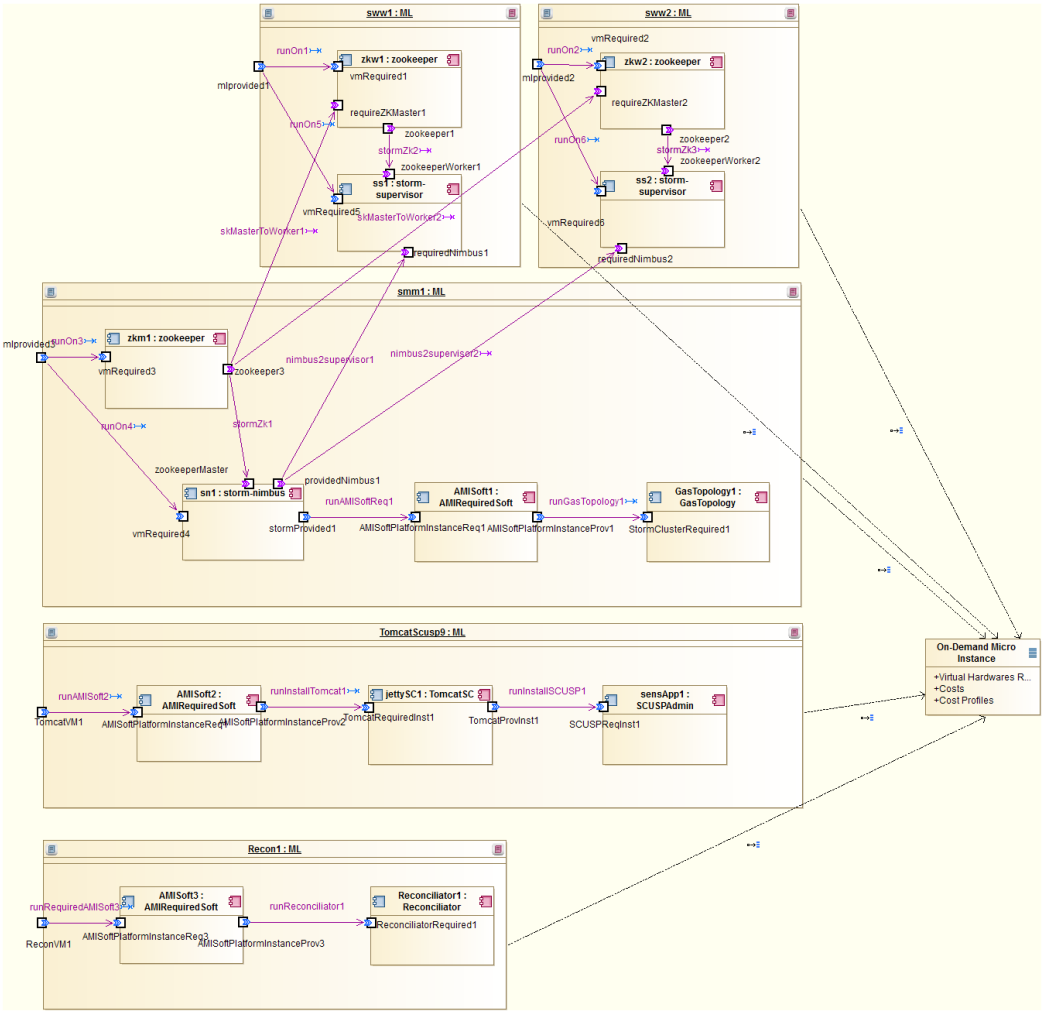


Figure 4: SCUSP' deployment model for Amazon EC2

3.1.2 Planning regarding upcoming tools evaluation

As we have stated earlier in the section "Context and Objectives" SCUSP use case consider necessary to use and coordinate with tool partners regarding two topics:

- Data storage and data migration, we will use Cassandra[1] as data storage for our system. In the cloud context, we will use a database as service (DBaaS) provider that will provide us this functionality. This choice was based on two properties of CAP theorem[2] that should hold for our system: scalability and fault tolerance.
- Platform monitoring and its implications in decision support for instance management

Regarding a database, CAP theorem states that a distributed system cannot simultaneously provide all of the three guarantees:

- Consistency: all clients see the current data regardless of changes
- Availability: despite node failures, the system continues to work as expected

[1] http://cassandra.apache.org/

[2] http://en.wikipedia.org/wiki/CAP_theorem

- Partition tolerance: despite network or messages failure, the system continues to operate as expected

Currently in this database we will store aggregated data from traffic sensors. The dataset was downloaded from Open Data Aarhus[3] site
The dataset represents historical traffic data captured from more than 400 traffic sensors during a period of five months and will serve as input for the MODAClouds migration tool, namely Hegira4Cloud. Since we are expecting that this database to grow and to store a big volume of data, we believe that it is unfeasible to download the data to a different cloud provider on the fly. Therefore, this tool will help us to migrate the data when, for some unpredictable reason, we have to move to a different cloud service provider.
In the appendix you can find the data model of this dataset and commands for creating the necessary tables on Cassandra.
Regarding monitoring aspects, SCUSP use case presents some elements specific for area of application and technological implementation in the real world: potential disruption performance computing needs and large fluctuations regarding volume of data used or necessary to be communicated. Therefore source for monitoring will be specific log files, usage of processor and memory and data exchanged volume. Of course, this list may be extended during effective implementation phase.

[3] http://www.odaa.dk/

Chapter 4

Conclusions

This document, combined with the deliverable D8.5.1, presents the architectural design of the Smart City Safety Urban Safety Planner case study. It will serve as the main foundation for the case study implementation and evaluation activities that will be performed during the last year of the project.
Being built from the scratch, as the project evolves, it is possible that further adaptations and changes will be applied at the architectural level. These updates will be described in the document that will accompany the delivery of the final prototype.

Appendix A

Case Study application design

A1. Gas pipeline data model

In order to design the gas pipe isolation scenario, the following entity relationship diagram was sketched. Effective data store model will be delivered in D9.5.2.

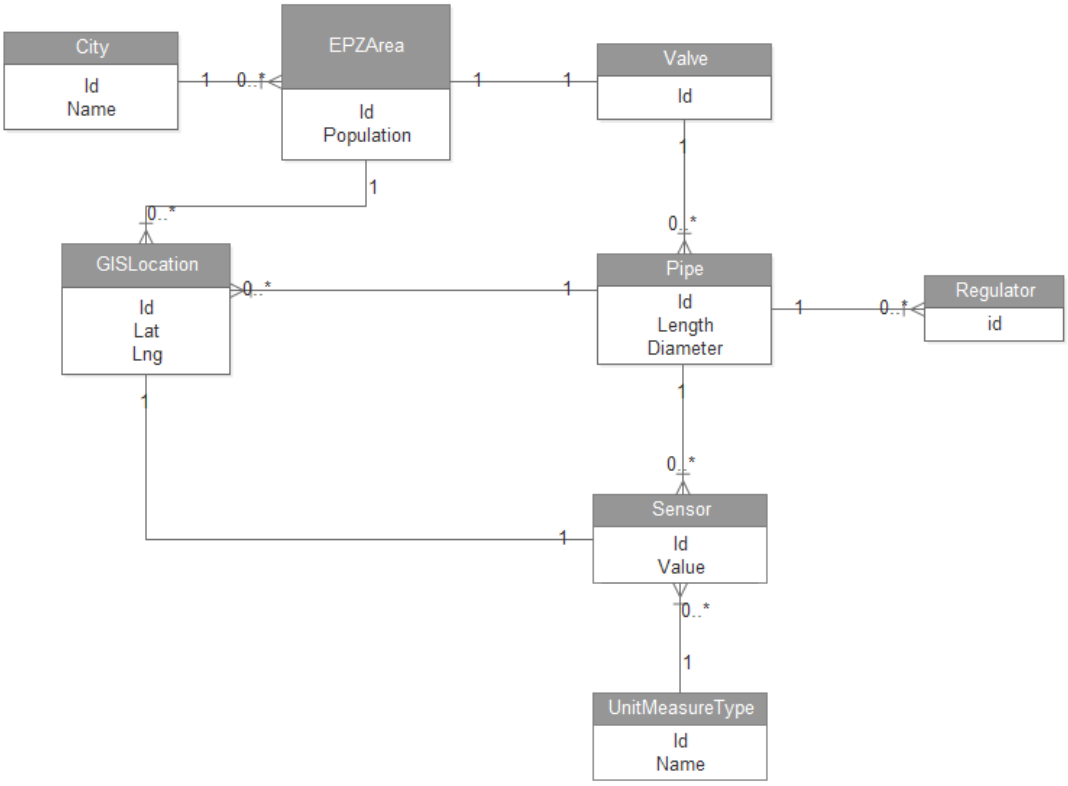


Figure 5: Entity relationship diagram for the gas pipe isolation

Entities

City: a relative large and permanent human settlement

EPZArea: Emergency Planning Zone is a hydraulically isolated area, which is a region that can be fully shut off to prevent feeding flammable gas into a pipeline incident

Valve: Any of various devices that regulate the flow of gases, liquids, or loose materials through piping or through apertures by opening

Pipe: A hollow cylinder or tube used to conduct a liquid, gas, or finely divided solid.

Regulator: A device used to control the flow of gases, liquids, or electric current

Sensor: A mechanical device sensitive to gas that transmits a signal to a measuring or control instrument.

UnitMeasureType: The unit measurement used for different values (e.g. ppm)

GISLocation: An Earth location defined by latitude, longitude and elevation

Relationships

A *City* has many *EPZArea*
An *EPZArea* is served by a single *Valve*
An *EPZArea* is bounded by many *GISLocation*
A *Valve* is covering many *Pipes*
A *Pipe* is monitored by many *Sensors*
A *Sensor* is placed in a one *GISLocation*
A *Pipe* has installed many *Regulators*
A *UnitMeasureType* can be allocated to many *Sensors*

A2. Traffic data model

Data from sensors are kept in two tables:

- traffic_sensors containing the metadata for sensors (such as locations and their identifier)
- traffic_records containing the vehicles count at different timestamps for each of the parts covered by the sensors

Below are the required commands for creating the corresponding tables for Cassandra:

```
CREATE TABLE traffic_sensors (POINT_1_LAT double,POINT_2_LAT double,
 POINT_1_LNG double,
 POINT_2_LNG double,
 DISTANCE_IN_METERS int,
 REPORT_ID int,
 POINT_1_NAME  varchar,
 POINT_2_NAME varchar,
 REPORT_NAME  varchar,
 PRIMARY KEY (REPORT_ID));

 CREATE TABLE traffic_records (
 status varchar,
 avgMeasuredTime float,
 avgSpeed float,
 extID int,
 medianMeasuredTime float,
 event_time timestamp ,
 vehicleCount int,
 u_id int,
 REPORT_ID int,
 PRIMARY KEY (REPORT_ID, event_time));
```

The dataset will be delivered as two separated CSV files.

A3. Telegram format

For our external data sources that are sending the events through the system we have defined the message exchange format as a JSON[4] message. Till now we have structured two types of telegrams: emergency and gas sensor.

A3.1 Emergency event telegram

In the context of this prototype we have defined the following types of emergency events:

[4] http://json.org/

FIRE, AMBULANCE, POLICE, GAS_LEAK, POWER_OUTAGE, OTHER. They are covered by the *eType* attribute. The location of the event is covered by the *lat* and *lng* attributes. Time of the event is captured by the *timestamp* attribute. The format of the telegram is depicted by the bellow structure:

```
{
        "eType": "FIRE",
        "lat": 45.63236088,
        "lng": 25.60254335,
        "description": "Call received",
        "timestamp": 1400135825444,
        "clazz": "EmergencyEvent"
}
```

A3.2 Gas Sensor telegram

The sensor telegram is depicted by the bellow telegram structure:

```
{
        "sensorId": "sens_40",
        "lat": 45.63236088,
        "lng": 25.60254335,
        "value": 84,
        "time": 1400135825220,
        "clazz": "CH4Event",
        "pipeId": 2
}
```

A4. Modelling SCUSP architecture with MODAClouds IDE

This part contains some architectural aspects as they were designed with MODACloudsIDE.

A4.1 Type level deployment diagram

Figure 6 illustrates the Type Level diagram:

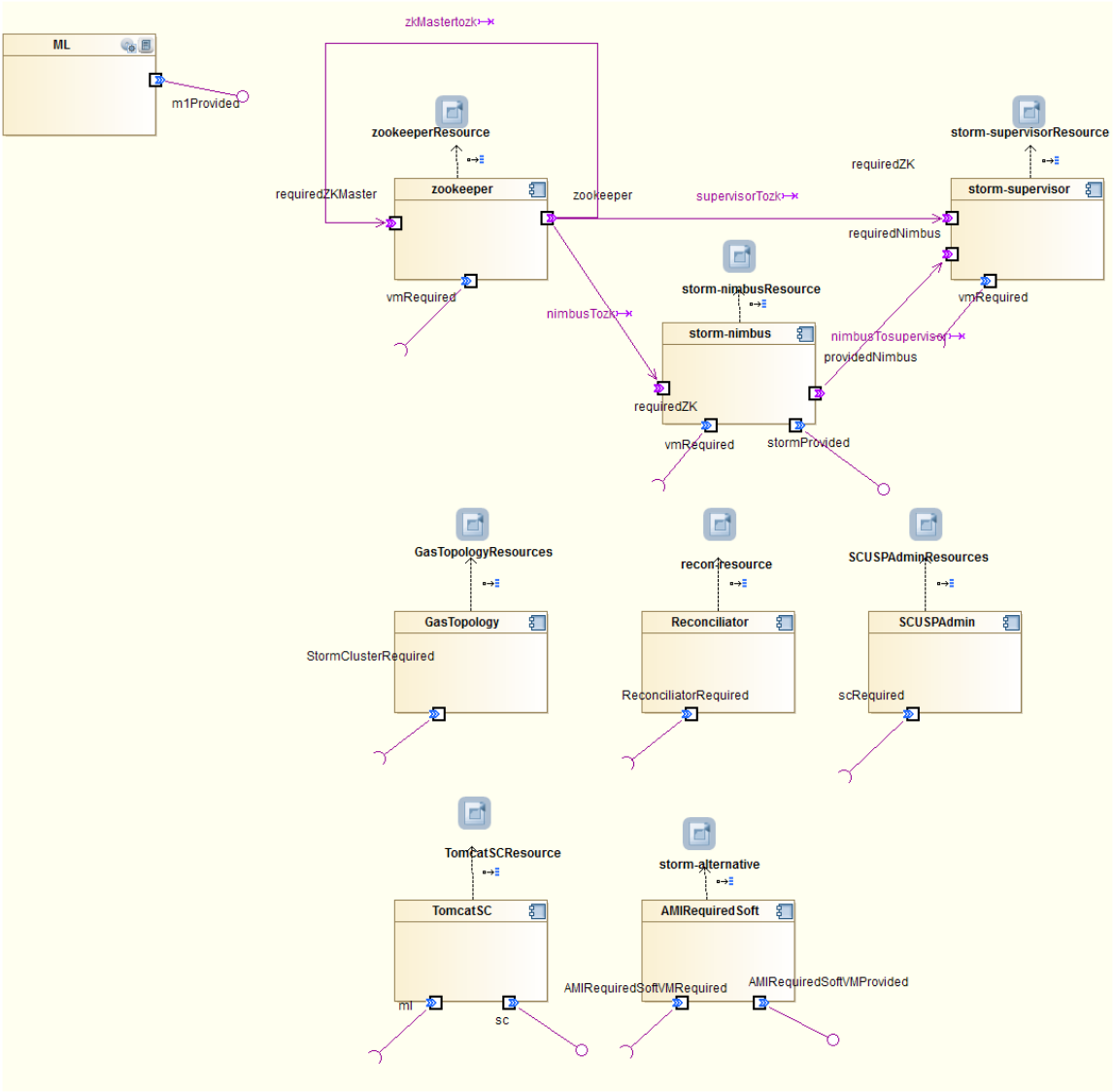


Figure 6: Type Level Diagram

A4.2 Deployment model at CSPM level for Flexiant provider

The deployment on Flexian Cloud provider will cover only the streaming engine, meaning the Storm cluster. Figure 7 illustrates the architecture at the CPSM:

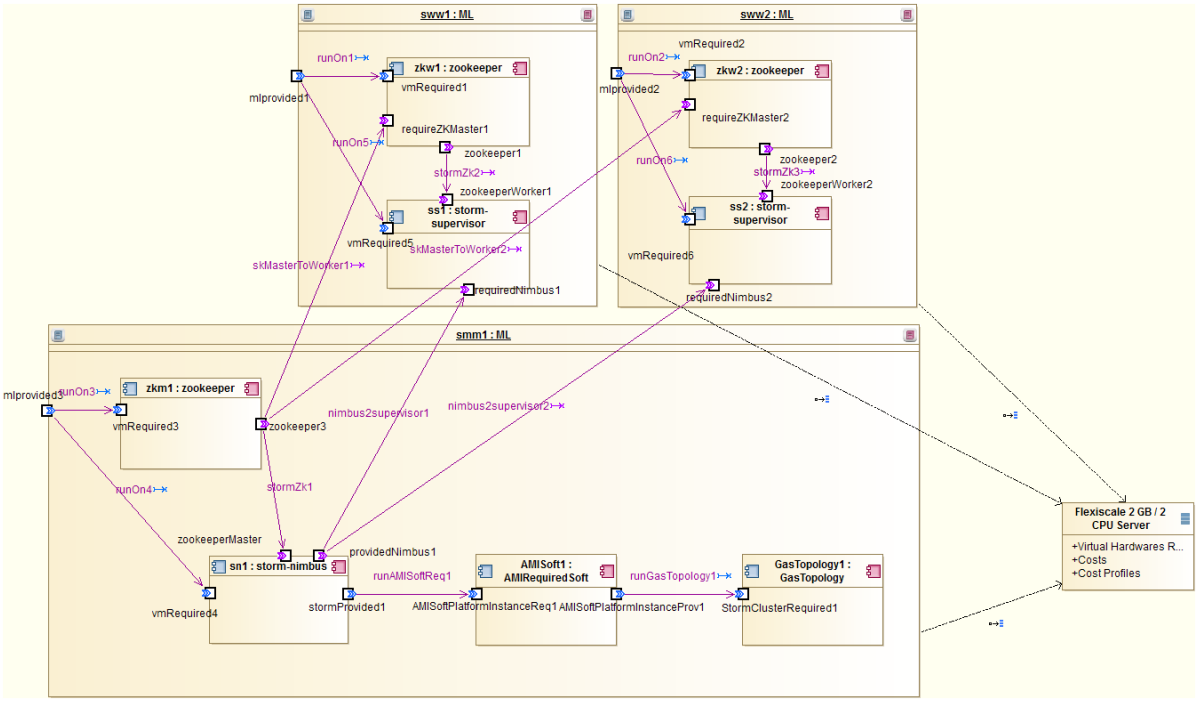


Figure 7: Deployment model for Flexiant