Katholieke
Universiteit
Leuven

**Department of
Computer Science**

Shared Internet Of Things Infrastructure Platform:

Domain Analysis

Software Architecture (H09B5a and H07Z9a) – Part 1

Filipcikova-Halilovic

Monika Filipcikova (r0683254)
Armin Halilovic(r0679689)
Academic year 2016–2017

Contents

1. Domain analysis

1.1 Domain models

The domain model is shown in figure **??**.

1.2 Domain constraints

In this section we provide additional domain constraints.

- Gateways can to only run light-weight programs that are not too demanding in resources.
- The SIoTIP Online Service must be almost permamently available and sensor data is never lost or inconsistent.
- Core running inside SIoTIP must be able to interact with external applications and systems.
- Relaying information to the motes by the gateway is done by GET and PUT requests.

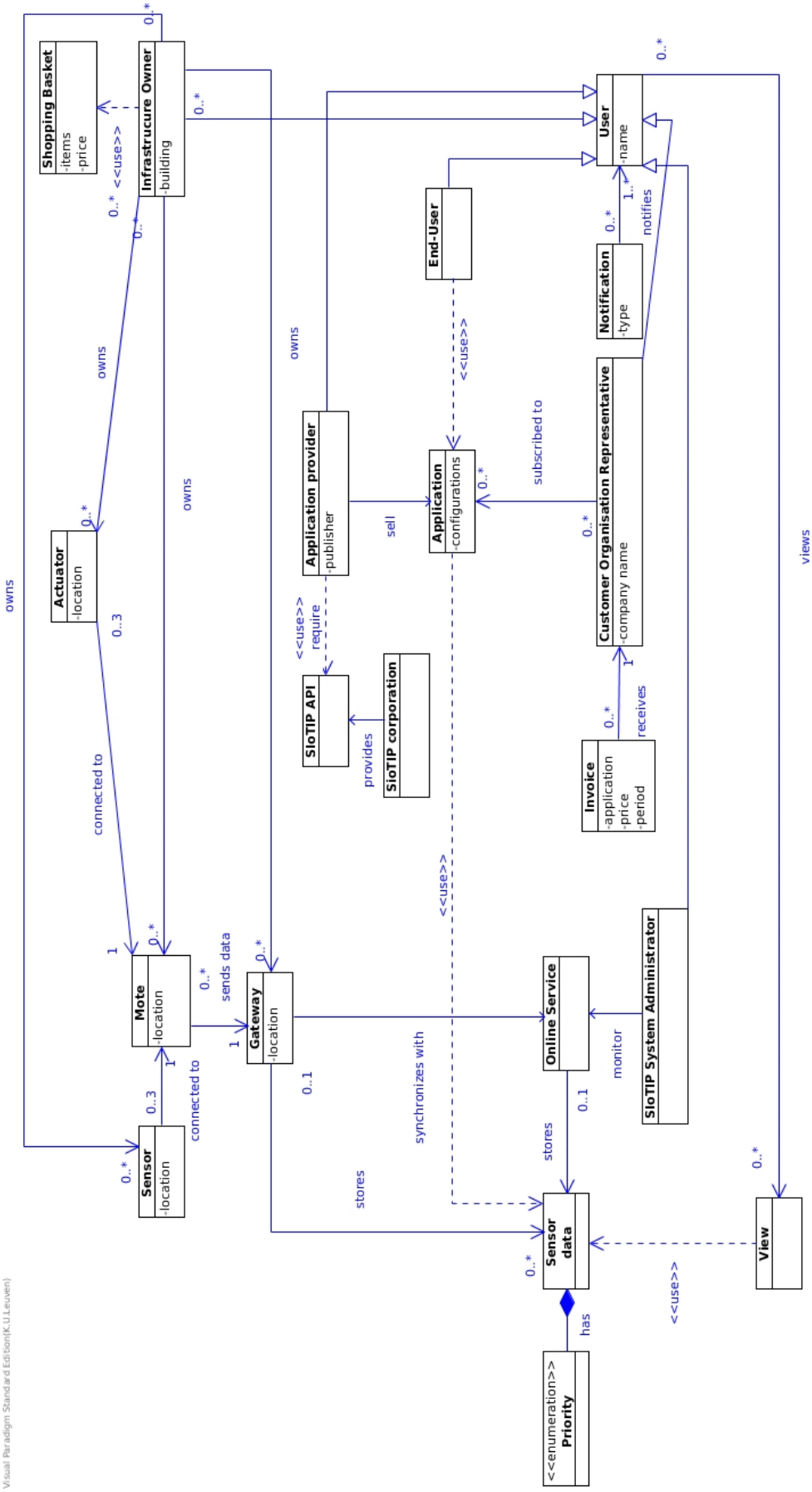

Figure 1.1: The domain model for the system.

1.3 Glossary

In this section, we provide a glossary of the most important terminology used in this analysis.

- **Application**: An application in the system is a program that uses sensors for a specific goal (e.g. fire detection). Customer organisations can subsribe to applications.
- **Application providers**: Application providers develop, upload, and maintain applications for the SIoTIP system.
- **Customer organisations**: Organisations that want to subscribe to applications. These organisations can subscribe to or unsubscribe from applications and get an overview of the invoices for their subscriptions.
- **Dashboard**: A dashboard is a collection of information which is available to a user. There are different dashboard for different type of users.
- **End-user**: End-users are the users that interact with deployed applications. They can issue commands to the application via a plethora of devices. For all end-users, the use of applications should simplify or automate their daily tasks.
- **Gateway**: A device which is connected to the Online Service and that has access to sensors and actuators. It can run some application logic and transmit data between the Online Service and the sensors/actuators.
- **Infrastructure owner**: This is someone who owns building(s), gateways, motes, sensors, and actuators. Owners are responsible for buying, installing, and maintaining hardware and the topology of their infrastructure. Owners can allocate installed hardware to specific customer organisations.
- **Mote**: A device that can host sensors and actuators. It connects these to the network and is used for communication between sensors/actuators and a gateway.
- **Notification**: A message that contains information about some event that occured in the system. Notifications have some priority. This influences how some notifications will be handled compared to others.
- **Online Service**: The online back-end which provides those services that are expected to generate the main revenue. Those services include the selling of hardware required for IoT infrastructures, and a platform where application providers and potential buyers can meet for a low price.
- **Sensor data**: Data about some sensor reading(s). Sensor data has some priority. This influences how some data will be handled compared to other data.
- **System**: "The system" is used to refer to the collection of entities that will be used by the SIoTIP corporation to render services. This includes the Online Service, gateways, motes, applications, etc.
- **System administrator**: System administrators monitor and manage the entire platform. They can view information about performance, running applications and sensors, shut applications down. These users are notified if alarming events threaten the platform.
- **User**: Someone who can log in and use parts of the system. Possibilities are system administrators, infrastructure owners, customer organisations, application providers, end-users.

2. Functional requirements

2.1 Use case model

Figure 2.1: Use case diagram for the system.

2.2 Use case overview

UC01: manage application A SIotIP system administrator manages an application.

UC02: review application A SIoTIP system administrator reviews an application and approves or declines it.

UC03: create account A SIotIP system administrator creates a user account (e.g. an Infrastructure Owner account).

UC04: upload application An appplication provider uploads an application to the Online Service. The application is automatically tested. If the application passes all tests, the application becomes available to customer organisations.

UC05: update application An application provider updates one of his applications. For example, the application provider wants deploy a new functionality or fix some bug.

UC06: debug application An application provider debugs one of his applications using the debugging environment provided by the system.

UC07: allocate hardware to customer organisation An infrastructure owner allocates installed hardware to specific customer organisations. This hardware can then be used by applications the customer organisations subscribe to.

UC08: get upload progress An application provider follows the progress of the upload process of an application.

UC09: browse applications A customer organisation representative browses the available applications through his dashboard.

UC10: subscribe to applications The customer organisation representative wants to add a new application. He has to send a request to the infrastructure owner.

UC11: unsubscribe from application A customer organisation unsubscribes from an application.

UC12: get invoices for applications The customer organisation gets an overview of the invoices for all of their application subscriptions.

UC13: configure application A customer organisation representative configures an application.

UC14: manage user roles A customer organisation representative manages the roles of users of applications the organisation is subscribed to.

UC15: notify A notification is sent to a user in the system. This notification contains data about an event that occured.

UC16: send heartbeat A mote sends a heartbeat to its gateway. This happens periodically.

UC17: send sensor data A mote sends data from a connected sensor to the gateway the mote is connected to.

UC18: get data from sensor An application requests to get data from sensors/actuators.

UC19: set data An application sends a command to set data at some hardware (gateways, sensors, or actuators).

UC20: log in A user authenticates himself in the system.

UC21: log out A user logs out of the system.

UC22: view notification A user views a list of his notifications.

UC23: buy new hardware An infrastructure owner orders new hardware through his dashboard. He adds new hardware to his shopping basket and confirms his purchase.

UC24: contact hardware manufacturer An infrastructure owner contacts a hardware manufacturer. For example, this could be to request integration of new hardware with already existing hardware.

UC25: manage topology of hardware An infrastrucutre owner adds, edits or removes information about the topology of sensors and actuators.

UC26: provide customer organisation data for registration An infrastructure owner adds customer organisation data (that is neccessary for registration) into the system.

2.3 Detailed use cases

2.3.1 *UC02*: Review application

- **Name:** Review application
- **Primary actor:** SIoTIP System Administrator
- **Secondary actor(s)**:
- **Interested parties:**
  - *application providers:* want to add their application to the system
- **Preconditions:**
  - The system administrator is authenticated.
  - The application needs to be reviewed by a system administrator.
- **Postconditions:**
  - The system adminstrator accepts or declines the application.
  - The application providers have been notified of this event.
- **Main scenario:**
  1. The system administrator navigates to the applications component on their dashboard and selects the application that needs to be reviewed.
  2. The system administrator reviews the application's functionality and log history, and the message history with the application provider.
  3. The system administrator indicates he wants to accept the application.

4. The system logs this event and makes the application available to customer ogranisations.
5. The system notifies the application providers.

- **Alternative scenarios:**

3b. The system administrator indicates he wants decline to the application and is prompted by the system to fill in the reason for this. The application will thus not become available to customer organisations.

3c. The system administrator indicates he wants communicate with the provider before making a decision. Afterwards, return to step 2.

- **Remarks:**
  - When an application first gets uploaded to the system, it needs to be reviewed and accepted by a system administrator. This will prevent the available apps to be flooded with apps of very poor quality or apps that are copies of other apps.
  - If at some point in the future, automated testing for an app fails, the app will need to be reviewed again before it can be made available again.

2.3.2 *UC06*: Upload application

- **Name:** Upload application
- **Primary actor:** application provider
- **Interested parties:**
  - *Customers organisations:* want to subscribe to the applications.
- **Preconditions:**
  - The application provider is authenticated.
  - The application provider has an application to upload.
- **Postconditions:**
  - The application is uploaded into the system.
  - The application passed the system's automated tests.
  - The application is made available to customer organisations for subscription.
- **Main scenario:**
  1. The application provider navigates to his dashboard.
  2. The application provider indicates he wants to upload an application.
  3. The application provider selects the application to be uploaded and uploads it.
  4. The system receives the application and stores it.
  5. The application undergoes a number of automated tests.
  6. The application provider follows the progress and results of these tests via the application provider dashboard.
  7. The application passes all tests.
  8. The system makes the application available for the customers organisations.
  9. The system sends a notification to the application provider.
- **Alternative scenarios:**

4b. The system did not receive the application and sends an error to the application provider. Return to step 2.

7c. The application did not pass all tests.

8c. The application will will need to be approved by a SIoTIP system administrator before it can be made available to customer organisations. (cf. review application (UCX))

2.3.3 *UC10*: Subscribe to application

- **Name:** Subscribe to application
- **Primary actor:** Customer orgranisation representative
- **Interested parties:**
  - *Infrastructure owner:* provides hardware that the application can use.
- **Preconditions:**
  - The customer organisation is authenticated.
  - The application is available to be subscribed to.
- **Postconditions:**
  - The customer organisation is subscribed to the application.
  - The customer organisation can use/configure the application.
- **Main scenario:**
  1. The representative indicates that he wants to subsribe to an application.
  2. The representative is notified that the application will be activated once the required hardware is installed.
  3. The infrastructure owner is notified of the subscription.
  4. The infrastructure owner updates the access rights to the hardware that is neccessary for the application.
  5. The application is activated.
  6. The customer organisation representative is notified.
- **Alternative scenarios:**

  2b. Extra hardware is required for the application to work.

  3b. The infrastructure owner is notified of the subscription and approves the purchase.

  4b. The infrastructure owner installs the new hardware and updates the topology and access rights.

2.3.4 *UC17*: Send sensor data

- **Name:** Send sensor data
- **Primary actor:** Mote
- **Secondary actors:** Sensor, Gateway
- **Interested parties:**
  - *Customer organisation:* pays for an application that uses this sensor.
  - *End-users:* use the sensor data in an application.
  - *Gateway:* stores this data.

- *Online Service:* stores this data.

- **Preconditions:**
  - The mote has received data from a sensor connected to it.
  - The mote is connected to a gateway.
- **Postconditions:**
  - The gateway has received and processed the sensor data.
  - The Online Service has received the data and can process it.
- **Main scenario:**
  1. The mote sends the sensor data to the connected gateway.
  2. The gateway receives the data and if applicable, runs some application logic.
  3. The gateway collects data until a synchronisation point is reached. At that point, the gateway sends the data to the Online Service.
- **Alternative scenarios:**

  3b. The gateway determined that the data was important (e.g. cause for alarm, notification, etc.) and sent the data to the Online Service immediately instead of waiting for the sycnhronisation point.
- **Remarks:**
  - It is essential that the synchronisation protocol works correctly in the presence of non-reliable network communication so that there is no loss of data.

2.3.5 *UC18*: Get data from sensors/actuators

- **Name:** Get data from sensors/actuators
- **Primary actor:** Application
- **Secondary actor(s)**: Online Service, Gateway, Mote
- **Interested parties:**
  - *End-user:* wants to get information from sensors.
- **Preconditions:**
  - The application is enabled.
  - The hardware that the application tries to get data from is connected to the system and can be used.
  - The application has access rights to the hardware it tries to get data from.
- **Postconditions:**
  - The application received the data it requested.
- **Main scenario:**
  1. The application sends a request to the Online Service, specifying which hardware it wants data from.
  2. The Online Service communicates this data to the relevant gateways.
  3. The gateways send requests for data to motes.
  4. The motes send data to the Online Service (cf. send sensor data (UCXX))
  5. The Online Service collects all the data and sends it to the application.
- **Alternative scenarios:**

  5b. The application recognizes it did not receive any reply within 1s. Return to step 1.

3. Non-functional requirements

In this section, we model the non-functional requirements for the system in the form of *quality attribute scenarios*. We provide for each type (availability, performance and modifiability) one requirement.

3.1 Availability

3.1.1 *Av1*: Unusable database

A database in the system cannot be used anymore. This could happen because of corrupted disks, power outages, too many requests coming into the database server, network failures, natural physical disasters, etc. A different database will be used while this one is unusable.

- **Source:** Database server
- **Stimulus:**
  - The database crashed.
  - The database does not respond to requests.
  - The database does not respond to requests within reasonable time.
  - The database returns invalid data or responses.
- **Artifact:** Persistent storage
- **Environment:** Normal operation
- **Response:**
  - Use a working replica until the database can be used again.
  - Log the fault.
  - If the problem with the database cannot be fixed automatically (e.g. by an automatic restart and resynchronisation), send a technician if it is possible for them to fix the problem (e.g. replace corrupted disks). Otherwise (e.g. power outages, natural physical disasters), send a notification of high priority to SIoTIP system administrators.
- **Response measure:**
  - If a database becomes unusable, this should be detected within 3s of the system trying to use the database.
  - When the system detects that a database is unusable, a working replica should be used within 5s.
  - If a database is unusable because of a system crash or user error, the database should restart and start its boot procedure within 5s.
  - If a database is unusable because of corrupted disks, a technician should replace the disk within 30 min.

3.1.2 *Av2*: Broken sensor

A sensor breaks. The infrastructure owner is notified and if possible, another sensor is used for the responsibility that the broken one was fulfilling.

- **Source:** Sensor
- **Stimulus:**

  - No data can be received anymore from the sensor.
  - The sensor stopped sending regular updates.
  - The sensor sends invalid/corrupted data.
  - The sensor disappeared from the heartbeats of the mote it is connected to.
- **Artifact:** Communication channel between sensor and gateway
- **Environment:** Normal operation
- **Response:**
  - The gateway uses another sensor to be used for the same responsibility as the broken one.
  - Log the fault and notify the infrastructure owner.
- **Response measure:**
  - When a sensor breaks, this is detected within 20s.
  - If there is another sensor that can fulfill the same responsibility as the broken one, it should be chosen within 5s.

3.2 Performance

3.2.1 *P1*: Application requests under peak load

An application sends requests to the Online Service while the system is under peak load. These requests could be for getting data from a specific sensor, for configuring application settings, etc. These request should be processed in a timely manner.

- **Source:** Application
- **Stimulus:**
  - An application sends a request to the Online Service.
- **Artifact:** The whole system
- **Environment:** Under peak load
- **Response:**
  - When a request comes into the system, core work necessary to generate a reply is done first. Other work of less importance (e.g. sending emails, generating low-priority notifications, etc.) is scheduled to be done later.
  - Load balancing is used to divide the requests over available servers.
  - Servers that are closest to source of the request are chosen over servers that are farther away to minimize network delay.
- **Response measure:**
  - Applications get a response within 3s.

3.2.2 *P2*: Mote data to application delay

When a mote sends data to an application, that data reaches its destination in a bounded time. This bound is determined by the priority of the data. The possible priorities are low, medium, and high. These priorities can be set by an infrastructure owner.

- **Source:** Mote
- **Stimulus:**
  - A mote sends data to an application.
- **Artifact:** The whole system
- **Environment:** Normal operation
- **Response:**
  - The data is always sent to the next node before other extra work is done. For example, other work could be logging or sending notifications.
- **Response measure:**
  - If the data priority is low, the data reaches the application within 60s.
  - If the data priority is medium, the data reaches the application within 10s.
  - If the data priority is high, the data reaches the application within 1s.

3.3 Modifiability

3.3.1 *M1*: Add a new type of sensor

The SIoTIP corporation wishes to provide a new type of sensor to customers.

- **Source:** SIoTIP developer
- **Stimulus:**
  - Developers add a new type of sensor to the system
- **Artifact:** System codebase and database.
- **Environment:** Normal operation, during a development iteration
- **Response:**
  - The software on motes and gateways must be updated so that the new sensor can be read, calibrated and configured.
  - Components in the system responsible for storage of sensor readings must be updated to save the readings of the new sensor correctly.
  - UI components must be updated so that the readings of the new sensor can be displayed correctly.
  - This modification does not affect the way data is communicated whithin components of the system.
  - Infrastructure owners can choose when the update will happen. The motes and gateways must be updated before the new typ of sensor can be ordered.
  - The updates to the UI components and storage components can be deployed at run time. There is no need to destroy any login sessions for this, since the sensor will not be in use at that time yet.
- **Response measure:**
  - This modification is finished within 5 man months of development time.
  - The deployment of the modifications to the UI components and storage components is finished within 10 minutes.

3.3.2 *M2*: Changes to UI components or new UI components

Developers want to add or edit components that make up the UI. For example, they want to improve components that are reused in dashboards.

- **Source:** SIoTIP developers
- **Stimulus:**
  - Developers add a UI component.
  - Developers edit a UI component.
- **Artifact:** User interface and platform.
- **Environment:** Normal operation, during a development iteration
- **Response:**
  - Users are not able to use the system during the deployment of the changes. All users that are logged in at the time the deployment starts will be logged out first.
  - All users are notified in advance of incoming downtime.
  - The update does not affect currently ongoing activity in the system.
  - The update only affects the subsystem responsible for the UI.
- **Response measure:**
  - The deployment of the new UI components is finished within 30 minutes. Users can log in again immediately after the deployment is finished.

3.4 Usability

3.4.1 *U1*: System administrator reviews application

Before an application can be published or when its automated tests fail, it needs to be reviewed by a SIoTIP system administrator.

- **Source:** SIoTIP system administrator
- **Stimulus:**
  - The system administrator wants to review the application and/or test logs quickly.
  - The system administrator wants to contact the application provider.
- **Artifact:** System administrator dashboard
- **Environment:** At normal operation
- **Response:**
  - The system administrator dashboard is built up using clear components which contain informations about different elements of the system. One such component displays applications that are waiting for a review.
  - The application review page contains a component that describes the application's functionality. There are clear buttons to accept or decline the application. There is a button to open the application.
  - Optionally, the application review contains a component which displays images of the application from an infrastructure owner's point of view.
  - There is a component which displays the log history of the application.

  - There is a component for communication with the application provider.

- **Response measure:**
  - All actions for reviewing the application, accepting or declining the application, reading the application's log history, and contacting the application provider are possible within 5 clicks.

3.4.2 *U2*: User wants to learn to use the system

A user wants work with the system efficiently. He wants to learn to use the system by reading guides for tasks or by searching for specific infomation quickly. An easy to use help system is put in place for this.

- **Source:** User
- **Stimulus:**
  - The user wants to learn system features.
  - The user wants to feel comfortable with his dashboard.
- **Artifact:** Help subsystem, UI components
- **Environment:** At normal operation
- **Response:**
  - The user uses the help system to learn about the functionality every page. When this help system is used, it displays key information about the page the user is currently on.
  - The help system contains a search form so the user can learn about anything they want that is relevant to their dashboard.
  - For each task a specific type user can perform, the help system contains a guide that explains the process step by step.
  - The user's dashboard consists of clear UI components that provide information about different elements of the system.
  - All forms have clear error notifications. The user is navigated the right way when he does something wrong.
- **Response measure:**
  - Anything the user would want to know about relevant to the system can be learned within 5 clicks.