SIoTIP: Assignment Part 2a Attribute-Driven Design (ADD)

Software Architecture: 2016–2017 project assignment

1 Instructions

The project involves the creation of the SIoTIP software architecture. In this part (2 labs), you are asked to sketch an early draft of the architecture using the Attribute-Driven Design (ADD) process. Later in the course (part 2b), you will produce a complete architecture.

Supporting material: The Attribute-Driven Design (ADD) process is described in Chapter 17 of the book *"Software architecture in practice (Third edition)"*[1]. A large catalog of patterns is contained in the book "*Pattern Oriented Software Architecture, Volume 4 – A Pattern Language for Distributed Computing*". On Toledo you will find a set of slides explaining tactics, patterns and other pointers, as well as the ADD process. In addition, there is a supporting document that explains the application of ADD on the running example (patient monitoring system).
You are strongly advised to refer to and use this material.

Starting point: The requirements presented in this document serve as the starting point for Part 2a. Tables 1 and 2 show the intrinsic priority of the requirements (likelihood and impact already taken into account). You will need these priorities in the ADD process to select suitable architectural drivers. Note that these requirements are meant to be a starting point for Part 2a and are not necessarily complete with regard to the assignment of Part 1.

Assignment: Perform **two decompositions** (i.e. runs of the ADD process). Informally: the initial decomposition plus one additional zoom-in.

Priority	Quality requirement
High Priority	M1, Av3, U2
Medium Priority	Av1, Av2, P1
Low Priority	P2, U1, M2

Table 1: Priority of the SIoTIP quality requirements.

Priority	Functionality	Use cases
High	*Changing infrastructure and topology, Pluggable device data and commands, Heartbeats, Application activation/deactivation/subscription, Services towards applications*	UC4, UC5, UC6, UC7, UC8, UC9, UC10, UC11, UC12, UC14, UC17, UC18, UC19, UC20, UC25, UC26, UC27
Medium	*Pluggable devices configuration, Upload and update applications, Consult historical data, Login/Logout*	UC13, UC22, UC24, UC28, UC29
Low	*Registration, Notifications, Invoicing, Consult application information*	UC1, UC2, UC3, UC15, UC16, UC21, UC23

Table 2: Priority of the SIoTIP functional requirements.

[1] Chapter 7 in the second edition.

The report

In the report for part 2a you are asked to provide (1) a detailed description of your ADD execution, and (2) a standalone description of the resulting architecture after two decompositions (ADD runs).

1. ADD execution We expect you to keep a detailed log during the execution of the Attribute-Driven Design (ADD) process, documenting each decomposition in a different subsection. Apply the following naming convention for the title of these subsections: "`Module_that_is_decomposed` (`Architectural_Drivers`)" (e.g. "PMS System (Av1, UC4, UC5)"). Each time you complete a decomposition, we expect you to **clearly and concisely** document:

- The selected architectural driver(s) for the module you are decomposing (you can and are in fact encouraged to group architectural drivers in one ADD run if they are sufficiently related) and the reasons for selecting and grouping these (rationale for driver selection).
- A systematic explanation of how you solve these requirements; i.e. the key architectural decisions taken during this iteration. Provide UML diagrams to visualize these solutions: at least one component diagram, deployment diagrams when needed (e.g. to show replication).
  - If you have adopted tactics to address the quality requirements, mention explicitly **which tactic and why you picked it.**
  - If you applied architectural pattern(s), provide an explanation of how they support the selected tactics. Pay attention to instantiate the patterns correctly and document what each module of the decomposition does, in collaboration with the other modules.
- An overview of the alternatives (solutions, patterns, tactics, other) you considered, and the reasoning behind not adopting them after all.
- The requirements (quality and functionality) that are left and need to be considered for each module of the current decomposition, i.e., the result of the verify-and-refine step of ADD.

2. Resulting architecture This section should present the component diagram of the overall system (after two decompositions). At this point, you are not required to provide the deployment diagram of the overall system.

Formatting rules You can use the tool of your choice to author the report, but you must deliver a single PDF file, preferably with indexes enabled for easy navigation (easy to achieve in LaTeX with the hyperref package). No other digital formats are accepted. The file must be self-contained (e.g., no extra figures in attachment) and readable (e.g., font size in pictures is adequate). Pages must be numbered. The cover page must mention the course name and the team member names (including their student id numbers). If you decide to use LaTeX, you can use the template that has been provided (on Toledo).

For UML modeling, you are recommended to use *Visual Paradigm for UML* (available in the PC labs, and installable on your own machine).

Delivery The deadline to turn in your report for part 2a is **Friday March 24 (noon)**. You are expected to (i) upload a self-contained PDF document on Toledo, and (ii) deliver a printed copy of this report in project post boxes (in the student room A00.03) on the main floor of the Department of Computer Science.

Good luck!

The Software Architecture team.

Contents

2 Quality requirements

The quality requirements are documented in the form of *quality attribute scenarios*.

2.1 *Av1*: Communication between SIoTIP gateway and Online Service

The Online Service can no longer receive data from a registered SIoTIP gateway or a SIoTIP gateway can no longer reach the Online Service due to a failure of the intermediate channel, a failure of (an internal communication component of) the SIoTIP Online Service or a failure of (an internal communication component in) the SIoTIP gateway.

- **Source:** Internal/external
- **Stimulus:**
  - The communication channel between the SIoTIP gateway and the Online Service has failed.
  - The Online Service (or an internal communication component) has failed.
  - The SIoTIP gateway (or an internal communication component) has failed.
- **Artifact:** Online Service communication component, SIoTIP gateway or communication channel(s)
- **Environment:** Normal mode
- **Response:**
  - Prevention:
    - The SIoTIP corporation has negotiated a Service-Level Agreement (SLA) with all involved telecom operators:
      - The connection to the Online Service is available at least 99.9% of the time, measured per year.
      - The network provides a minimal data rate of 128KB/s.
    - SIoTIP demands from infrastructure providers that the local network through which the SIoTIP gateway connects to the Online System is available at least 90% of the time, measured per year.
  - Detection:
    - The Online Service is able to autonomously detect failures of its individual internal communication components.
    - The Online Service is able to detect that a SIoTIP gateway is not sending data anymore based on the expected synchronisation interval.
    - The SIoTIP gateway is able to autonomously detect failures of its individual internal communication components.
    - The Online Service should acknowledge each message sent by the SIoTIP gateway so that the gateway can detect failures.
  - Resolution:
    - The Online Service notifies the infrastructure manager and a SIoTIP system administrator when the outage of a SIoTIP gateway is detected.
    - If an internal SIoTIP gateway component fails, the gateway first tries to restart the affected component. If the failure persists, the SIoTIP gateway reboots itself entirely. Note that the SIoTIP gateway, due to the occurred failure, cannot contact a system administrator itself.
    - If (an internal communication component of) the Online Service or the communication channel has failed, the SIoTIP gateway will temporarily store all incoming pluggable data and any issued application commands internally.
    - If the Online Service becomes unreachable, application parts running locally on the SIoTIP gateway continue to operate normally.

- **Response measure:**
  - The failure of an internal SIoTIP Online Service component is detected within 30 seconds.
  - The detection time for a failed SIoTIP gateway or channel depends on the transmission rate of the gateway. An outage is defined as 3 consecutive expected synchronisations that do not arrive within 1 minute of their expected arrival time.
  - The infrastructure owner is notified within 5 minutes after the detection of an outage of their gateway (*UC15*: Send notification).
  - A SIoTIP system administrator should be notified within 1 minute after the detection of a simultaneous outage of more than 1% of the registered gateways.
  - The SIoTIP gateway will start synchronising with the Online service within 1 minute after the communication channel becomes available.
  - The SIoTIP gateway can store at least 3 days of pluggable data before old data has to be overwritten.

2.2 *Av2*: Application failure

The Online service or a SIoTIP gateway can no longer correctly execute applications due to a crash of an application or its container, or a failure of an internal application execution component.

- **Source:** Internal
- **Stimulus:**
  - An application executing on the Online Service has crashed
  - An application executing on a SIoTIP gateway has crashed
  - A container in which an application executes has crashed
  - An internal application execution component in the Online Service or SIoTIP gateway has failed.
- **Artifact:** Online Service application execution subsystem or SIoTIP gateway application execution
- **Environment:** Normal mode
- **Response:**
  - This does not affect other applications that are executing on the Online service or SIoTIP gateway.
  - This does not affect the availability of other functionality of the system, such as the dashboards.
  - Prevention:
    - Applications fail independently: they are executed within their own container to avoid application crashes to affect other applications.
    - The subsystem for executing applications in the Online Service must have a guaranteed minimal up-time.
  - Detection:
    - The system is able to autonomously detect failures of its individual application execution components.
    - The system is able to autonomously detect failing applications and application containers.
    - Upon detection, a SIoTIP system administrator is notified.
  - Resolution:
    - In case of application crash, the system autonomously restarts failed applications.
    - In case of failure of application execution components or an application container, a system administrator is notified.

    - If part of an application fails, the remaining parts remain operational, possibly in a degraded mode (graceful degradation).

- **Response measure:**
  - The failure of an internal application execution component is detected within 30 seconds.
  - The subsystem for executing applications in the Online Service must be available 99% of the time, measured per month.
  - Detection of failed hardware or crashed software happens within 5 seconds.
  - SIoTIP system administrators are notified within 1 minute.
  - After 3 failed restarts the application is suspended, and the application developer and customer organisation are notified within 5 minutes.

2.3 *Av3*: Pluggable device or mote failure

A pluggable device or mote fails or becomes unresponsive, thereby possibly impacting the active applications.

- **Source:** External
- **Stimulus:** A pluggable device or mote has failed.
- **Artifact:** SIoTIP gateway and Online Service application execution subsystem
- **Environment:** Normal mode
- **Response:**
  - Prevention:
    - Application providers can design their applications such that they explicitly require redundancy in the available pluggable devices (e.g. multiple smoke detectors in a room for fire detection) before an application can be activated. This allows applications to continue to operate normally even in the event of failure of a single pluggable device or mote.
  - Detection:
    - A SIoTIP gateway can autonomously detect failure of one of its connected motes and pluggable devices (cf. *UC14*: Send heartbeat).
  - Resolution:
    - The infrastructure owner should be notified of any persistent pluggable device or mote failures.
    - Applications that can no longer operate due to failure of a pluggable device or mote should be automatically suspended and re-activated once the failure is resolved.
    - Customer organisations should be notified if one or more of their applications is suspended or re-activated.
    - Applications using a failed pluggable device or any device on a failed mote should be notified.
- **Response measure:**
  - A failed mote is defined as 3 consecutive heartbeats that do not arrive within 1 second of their expected arrival time.
  - A failed pluggable device is detected within 2 seconds after the arrival of a heartbeat of the mote in which it was plugged in.
  - Infrastructure owners are notified within 1 minute after detecting a mote outage lasting at least 10 seconds.
  - Infrastructure owners are notified within 1 minute after the detection of the unavailability of a pluggable device for 30 seconds.

  - Applications are notified of the failure of relevant pluggable devices within 10 seconds.
  - Applications are suspended within 1 minute after detecting the failure of an essential pluggable device.
  - Application are reactivated withing 1 minute after the failure is resolved.

2.4 *P1*: Large number of users

Upon success, SIoTIP should be able to easily accommodate the needs of their increasing customer base which mainly consists of infrastructure owners and customer organizations. The SIoTIP platform should in other words be able to deal with increasing numbers of gateways, motes and pluggable devices that constantly share the obtained sensor data with SIoTIP. In addition, as the number of customer organisations increases, SIoTIP should be able to execute a larger amount of applications.

- **Source:** Infrastructure owner/Customer organisation/SIoTIP gateway
- **Stimulus:**
  - An infrastructure owner uses the SIoTIP services directly:
    - *UC4*: Install mote
    - *UC5*: Uninstall mote
    - *UC6*: Insert a pluggable device into a mote
    - *UC7*: Remove a pluggable device from its mote
    - *UC8*: Initialise a pluggable device
    - *UC9*: Configure pluggable device access rights
    - *UC10*: Consult and configure the topology
    - *UC16*: Consult notification message
  - A customer organisation uses the SIoTIP services directly:
    - *UC1*: Register a customer organisation
    - *UC2*: Register an end-user
    - *UC3*: Unregister an end-user
    - *UC16*: Consult notification message
    - *UC19*: Subscribe to application
    - *UC20*: Unsubscribe from application
  - or indirectly via their applications:
    - *UC12*: Perform actuation command
    - *UC13*: Configure pluggable device
    - *UC24*: Consult historical data
    - *UC25*: Access topology and available devices
    - *UC26*: Send application command or message to external front-end
    - *UC27*: Receive application command or message from external front-end
- **Artifact:** SIoTIP gateway and Online Service
- **Environment:** Normal mode
- **Response:**
  - The SIoTIP Online Service replies to the service requests of the infrastructure owner and customer organisations.
  - The Online Service processes the data received from the gateways.
  - The application execution subsystem should be able to execute an increasing number of active applications.

- **Response measure:**
  - The initial deployment of SIoTIP should be able to deal with at least 5000 gateways in total, and should be provisioned to service at least 3000 registered users simultaneously connected to SIoTIP.
  - Scaling up to service an increasing amount of infrastructure owners, customers organisations and applications should (in worst case) be *linear*; i.e. it should not require proportionally more resources (machines, etc.) than the initial amount of resources provisioned per customer organisation/infrastructure owner and per gateway.

2.5 *P2*: Requests to the pluggable data database

The internal database responsible for storing the pluggable data receives a large amount of parallel requests, e.g. for storing new pluggable device data or retrieving sensor data.

- **Source:** Internal subsystems
- **Stimulus:** Multiple internal subsystems send requests to the pluggable data database in parallel.
- **Artifact:** SIoTIP Online Service
- **Environment:** Normal mode
- **Response:**
  - In normal mode, the database processes incoming requests in a first-in-first-out order.
  - If the system fails to comply to the deadlines specified below, it goes in overload mode: requests are handled in the order that returns the system to normal mode the fastest, taking into account:
    - the nature of the requests: storing new pluggable data (*UC11*: Send pluggable device data) has priority over specific lookup queries (e.g. retrieving most recent measurement for a few sensors), which in turn have priority over broad queries (e.g. retrieving all sensor data for the last month).
    - the priority of an application: requests from applications marked as critical by their subscribers are processed before non-critical applications.
  - The processing of (large amounts of) requests concerning pluggable data has no impact on requests concerning other data, e.g. available applications.
  - Also a mechanism should be in place to avoid starvation of any type of request.
- **Response measure:**
  - Write requests are handled within 500msec.
  - Read requests from critical application are handled within 750msec.
  - Read requests from non-critical applications are handled within 1000msec.

2.6 *M1*: Integrate new sensor or actuator manufacturer

A new sensor or actuator manufacturer wants to integrate with SIoTIP. Such a manufacturer can provide pluggable devices similar to those already available (e.g. a temperature sensor measuring in degrees Fahrenheit instead of degrees Celsius) or provide completely new devices (e.g. a smoke detector). This should require minimal changes to the gateway software, data processing and storage in the Online Service.

- **Source:** A sensor or actuator manufacturer.
- **Stimulus:** The manufacturer wants to offer a new type of pluggable device supported by SIoTIP.
- **Artifact:** code, interfaces, processing and storage of pluggable data

- **Environment:** At design time or at run time
- **Response:**
  - This modification has the following consequences:
    * The new types of sensor or actuator data should be transmitted, processed and stored, and should be made available to applications.
    * The pluggable data processing subsystem should be extended with relevant data conversions, e.g. converting temperature in degrees Fahrenheit to degrees Celsius.
    * The available applications can be updated to use any new pluggable devices.
    * The infrastructure managers must be able to initialize the new type of pluggable device (*UC8*: Initialise a pluggable device), configure access rights for these devices (*UC9*: Configure pluggable device access rights), and view detailed information about the new type of pluggable device (*UC10*: Consult and configure the topology).
  - This modification must not affect:
    * The existing applications.
    * The hardware provided by the current sensor and actuator manufacturers.
- **Response measure:**
  - Applications should not require any code changes to use new pluggable devices equivalent or similar to one already in use.
  - Extending the pluggable data processing and storage facilities to handle any new types of actuator or sensor data should not take longer than 1 man week.
  - Application providers that develop a new application do not need to distinguish in any way between the pluggable device types that were already foreseen in the initial release of SIoTIP and the pluggable device types that were added at a later point in time.

2.7 *M2*: Big data analytics on pluggable data and/or application usage data

The availability of large amounts of pluggable data and information on the usage of applications offers interesting opportunities for big data analytics. The SIoTIP corporation can use this to improve their own services towards application providers and/or customer organisations. Furthermore, external third parties could, for a certain fee, get access to an anonymised copy of this data.

- **Source:** SIoTIP marketing department/External researchers
- **Stimulus:** The pluggable data and (application) usage statistics gathered by SIoTIP are used to improve its own services and are opened to external third parties, such as researchers.
- **Artifact:** SIoTIP subsystem storing pluggable data, SIoTIP monitoring application execution.
- **Environment:** At design time
- **Response:**
  - A new dashboard is introduced for external third parties who want to use SIoTIP's data.
  - The pluggable data storage subsystem must be extended with an API to perform big data analytics.
  - The increased number of queries to the pluggable data storage subsystem should not decrease its availability or performance with respect to queries form applications.
  - The application monitoring subsystem must be extended to offer fine-grained statistics concerning applications (e.g., used storage, CPU usage, number of end users, number and type of pluggable devices, number of active users, etc.).
  - Gathering statistics about applications or pluggable data usage should not interfere with the execution of applications.

- **Response measure:**
  - Developing, testing and deploying the new dashboard does not take longer than 1 man month.
  - Extending the pluggable data storage subsystem with a new API does not take longer than 6 man months.
  - Extending the application monitoring subsystem does not take longer than 6 man months.

2.8 *U1*: Application updates

Software upgrading and evolution is inevitable, to fix bugs, introduce new features, etc. This however should be as simple as possible from the point of the application provider.

- **Source:** Application provider
- **Stimulus:** Application providers need to be able to issue application updates to, for example, fix bugs or introduce new features.
- **Artifact:**
  - SIoTIP gateway
  - Application execution subsystem
  - Application provider dashboard
- **Environment:** At run time
- **Response:**
  - Application developers should be able to upload a new version of an application (cf. *UC22*: Upload an application) and SIoTIP should apply these updates to the affected active applications.
  - Application providers should be able to indicate the update policy to follow. For example, updates containing only bug fixes are free for all subscribers, whereas releases with new features require a subscription renewal.
  - The update mechanism should be robust: when an update fails, the previous application version should not be affected (still allowing roll-back).
  - The actual process of updating an application should be transparent from the point of view of the customer organisation (with the exception of renewing a subscription for releases with new features).
- **Response measure:**
  - Making a new application available to customer organisations should not require any human intervention other than uploading it (given that the uploaded application passes the automated tests).
  - Deploying updated applications should be fully automated, thus not requiring human intervention.
  - SIoTIP should be able to update one Online Service part of a single subscription within 10 minutes.
  - SIoTIP should have the capacity to push out application updates to at least 500 gateways in 5 minutes.
  - After receiving an application update, the application upgrade should not take longer than 5 minutes on a SIoTIP gateway.

2.9 *U2*: Easy installation

One of the key selling points of SIoTIP is that its practical installation requires minimal (to no) configuration by the infrastructure owner and therefore should be appealing to non-technical users as well.

- **Source:** Infrastructure owner, SIoTIP marketing department
- **Stimulus:** Infrastructure owner wants to install (new) hardware devices with a minimal configuration and installation effort.
- **Artifact:**
  - SIoTIP gateway and Online services
- **Environment:** At run time
- **Response:**
  - The gateway should not require any configuration, other than being connected to the local wired or WiFi network, after it is plugged into an electrical socket.
  - Installing a new mote should not require more configuration than adding it to the topology (cf. *UC4*: Install mote).
  - A removed mote that is reintroduced (with the same pluggable devices attached), is automatically re-added to the topology.
  - Adding new sensors or actuators should require no further customer actions besides plugging it into the mote (cf. *UC6*: Insert a pluggable device into a mote). Configurable sensors and actuators should have a working default configuration.
  - Applications should work out of the box if the required sensors and actuators are available.
  - Only when mandatory end-user roles must be assigned, additional explicit configuration actions are requirement from a customer organisation (cf. *UC17*: Activate an application and *UC19*: Subscribe to application).
- **Response measure:**
  - An infrastructure owner should be able get the SIoTIP gateway up-and-running (connected) within 10 minutes given that the information (e.g. WiFi SSID and passphrase) is available to the person responsible for the installation.
  - Adding new motes, sensors or actuators should not involve more than just starting motes, and plugging devices (sensors, actuators) into motes – plug-and-play!
  - Pluggable devices added to an already known mote are automatically added in the right location on the topology.
  - Making (initialised) sensors and actuators available to customer organisations and applications should not require more effort than configuring access rights (cf. *UC9*: Configure pluggable device access rights).

3 Functional requirements

This section describes the functional requirements for the SIoTIP system.

Actors

The actor hierarchy is shown in figure 1.

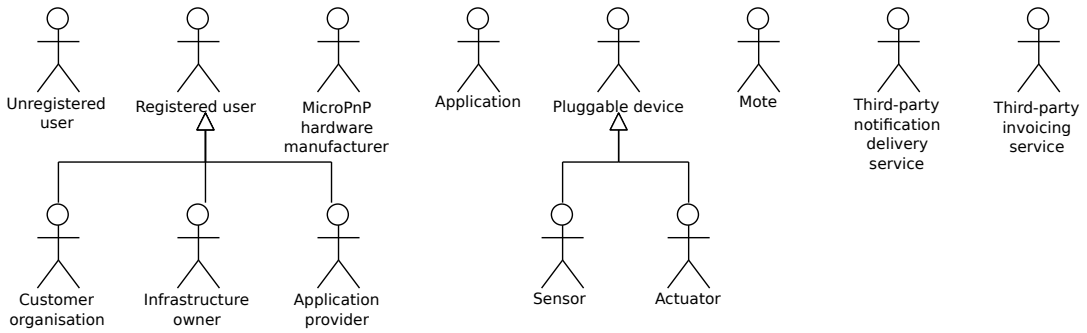

Figure 1: Actors involved in SIoTIP.

Use case diagram

The use case diagram is presented in figure 2.

3.1 *UC1*: Register a customer organisation

- **Summary:** An unregistered customer organisation is registered in order to use the SIoTIP applications.
- **Primary actor:** Unregistered user (as a representative of a customer organisation)
- **Secondary actors:** None
- **Interested parties:**
  - *SIoTIP:* wants as many registered customer organisations as possible.
  - *Unregistered user:* wants to use the applications available via SIoTIP.
- **Preconditions:** None
- **Postconditions:**
  - The primary actor now represents a customer organisation, is associated to an infrastructure owner, and can log in to the system on its behalf.
  - Notifications for the primary actor are delivered according to the selected delivery method (cf. *UC15*: Send notification).
  - End-users can now be registered that are associated with the newly-registered customer organisation. (cf. *UC2*: Register an end-user).
- **Main scenario:**
  1. The primary actor requests the (infrastructure owner-specific) registration form.
  2. The system asks the primary actor for an e-mail address.
  3. The primary actor enters a company e-mail address.

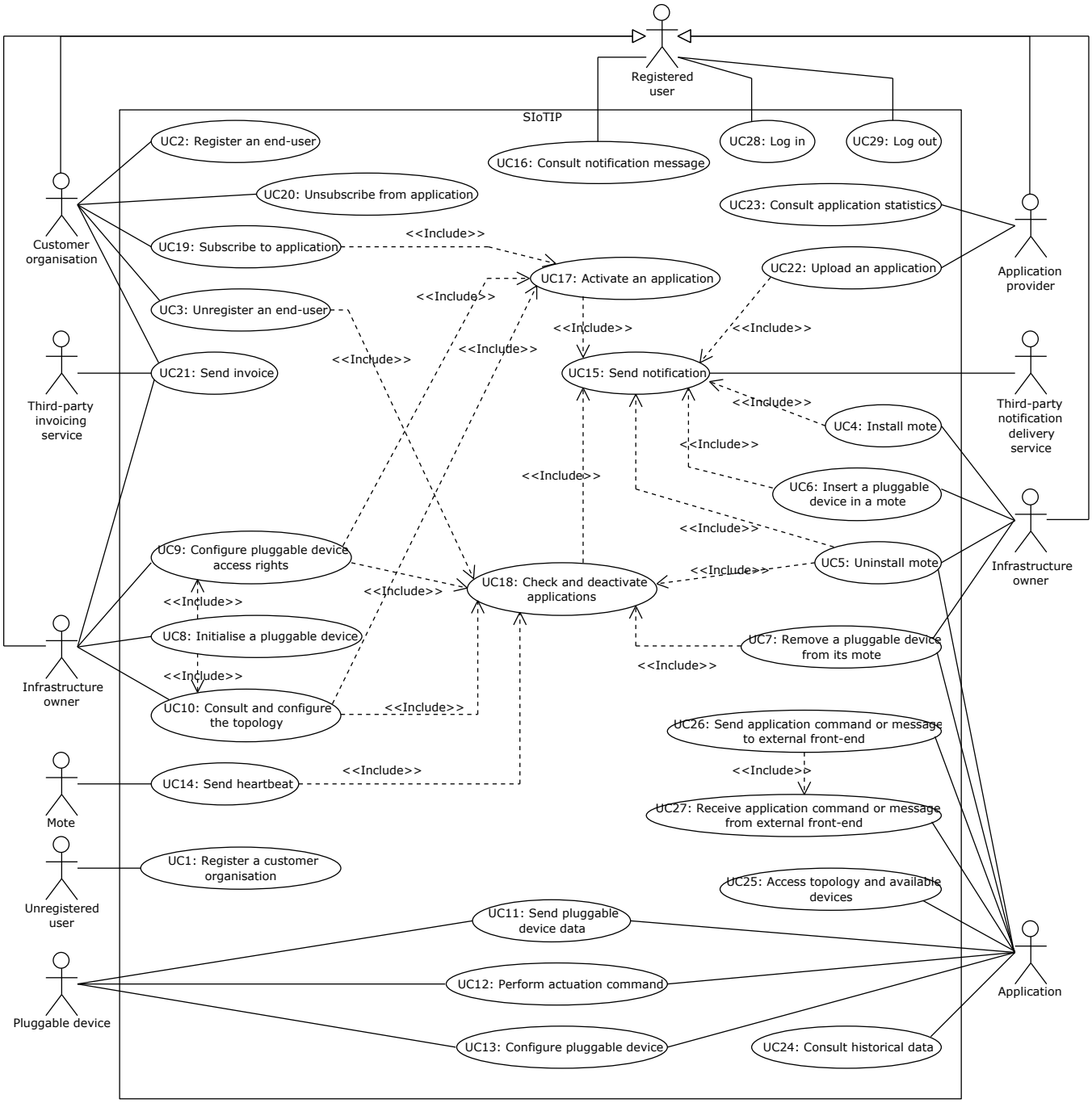

Figure 2: Use case diagram for SIoTIP.

4. The system verifies the provided e-mail address, and if the e-mail address is valid, the system informs the primary actor that a confirmation e-mail was sent to the entered e-mail address with further instructions.
5. The primary actor confirms his or her e-mail address by following the link in the received e-mail.
6. The system asks the primary actor for the remaining information:
   (a) Credentials (e.g., unique username and password)
   (b) Full name of the company and primary address
   (c) Billing and payment information (e.g. credit card information or bank account number, and, optionally, invoicing address)
   (d) Contact phone number
   (e) Notification preferences: SMS and/or e-mail.
7. The primary actor provides the requested information.
8. The system verifies the provided information, and, if the provided information is correct, the system creates a new customer organisation profile.
9. The system notifies the primary actor that the registration is successful.

- **Alternative scenarios:**

4a. If the provided e-mail address is already registered, the system informs the primary actor and the use case ends.

4b. If the provided e-mail address is not valid, e.g. badly formatted, the system informs the primary actor of this error. Continue from step 2.

5a. If the primary actor does not click the link in the received e-mail within 30 minutes, the use case is aborted.

8a. If the chosen user name already exists, the system asks the primary actor for a different user name. Continue from step 6.

8b. If the provided primary company address or invoicing address is not valid, e.g. the provided city and postcode do not match, the system asks the primary actor to correct this. Continue from step 6.

8c. If the provided payment information is not correct, e.g. incorrect credit card number, the system informs the primary actor of the encountered error. Continue from step 6.

- **Remarks:**
  - We assume there is a separate customer organization registration form for each infrastructure owner (e.g., a web form with a hidden field that contains the unique ID of the infrastructure owner), and that the primary actor has previously obtained the link to this form through some means outside the scope of the system.
  - The customer organisation can change this profile information later on, e.g. to change the contact phone number. For simplicity, no explicit use cases are provided for performing such changes.
  - A customer organisation can also be unregistered (triggered by the customer organisation itself, or by the associated infrastructure owner). This process involves unsubscribing from all applications (cf. *UC20*: Unsubscribe from application). For simplicity, no explicit use case is provided for this scenario.

3.2 *UC2*: Register an end-user

- **Summary:** An end-user is registered by a customer organisation in order to use SIoTIP.
- **Primary actor:** Customer organisation
- **Secondary actors:** None

- **Interested parties:**
  - *Customer organisation:* wants to allow some people (e.g., employees or contractors) to act as end-users in SIoTIP to use applications in order to increase business value.
  - *End-user:* wants to use the applications in SIoTIP.
- **Preconditions:**
  - The primary actor is logged in (cf. *UC28*: Log in).
- **Postconditions:**
  - The end-user is registered and associated to the customer organization, and can now be assigned roles in applications (cf. *UC19*: Subscribe to application).
- **Main scenario:**
  1. The primary actor indicates to the system that he or she wants to register an end-user.
  2. The system asks the primary actor for details about the end-user to be registered:
     - Name
     - Phone number (for SMS) and/or e-mail address
  3. The primary actor supplies the requested information.
  4. The system verifies the provided e-mail address or phone number, and checks that the end-user does not exist in the system already. If the provided information is valid, the system stores the end-user information and associates it with the primary actor.
  5. The system informs the primary actor that the end-user was registered successfully.
- **Alternative scenarios:**

  4a. If the provided phone number or e-mail address is not valid, e.g. badly formatted, or if the end-user for which details were provided already exists in the system, the system informs the primary actor of this error. Continue from step 2.
- **Remarks:** None

3.3 *UC3*: Unregister an end-user

- **Summary:** A registered end-user is unregistered by its associated customer organisation.
- **Primary actor:** Customer organisation
- **Secondary actors:** None
- **Interested parties:**
  - *Customer organisation:* does not want the end-user to use the SIoTIP applications for their organisation anymore.
- **Preconditions:**
  - The primary actor is logged in (cf. *UC28*: Log in).
- **Postconditions:**
  - The profile of the end-user is marked ‘inactive’ and the end-user can no longer be assigned roles in applications.
  - Applications in which the end-user was the only end-user to be assigned to some mandatory role (cf. *UC19*: Subscribe to application), are deactivated (cf. *UC18*: Check and deactivate applications).
- **Main scenario:**

1. The primary actor indicates to the system that they want to unregister an end-user.
2. The system retrieves all end-users associated with the primary actor and presents this list to the primary actor.
3. The primary actor indicates which end-user she wants to unregister.
4. The system marks the profile of the end-user as 'inactive'
5. The system checks if any applications should be deactivated because of end-user assignment to mandatory roles (**Include:** *UC18*: Check and deactivate applications).
6. The system informs the primary actor that the end-user is now unregistered from SIoTIP.

- **Alternative scenarios:**

  2a. If the primary actor has no associated end-users, the system informs the primary actor. The use case ends.

- **Remarks:** None

3.4 *UC4*: Install mote

- **Summary:** An infrastructure owner installs a new mote, allowing them to add new pluggable devices to the infrastructure.
- **Primary actor:** Infrastructure owner
- **Secondary actors:** None
- **Interested parties:**
  - *SIoTIP:* wants an extensive IoT infrastructure registered to their platform to support many applications and customer organisations.
  - *Infrastructure owner:* wants an infrastructure capable of supporting the needs of customer organisations, and requires the ability to overview and manage this infrastructure easily.
  - *Customer organisation:* wants to use the infrastructure to run applications to automate processes and increase business value.
  - *MicroPnP hardware manufacturer:* wants to sell as many motes as possible.
- **Preconditions:**
  - The primary actor has at least one associated gateway.
- **Postconditions:**
  - The new mote is associated with the gateway it is attached to.
  - The new mote is included in the primary actor's topology as an unplaced mote and can be further configured (cf. *UC10*: Consult and configure the topology).
- **Main scenario:**

  1. The primary actor physically activates the mote. (This could simply involve bringing it to the vicinity of the network, or may involve pressing a power button or physically plugging it in a power socket.)
  2. The system receives a broadcasted notification from the new mote, which includes the mote IP address.
  3. The system registers the mote by associating it to its gateway, and adds it as an unplaced mote in the primary actor's topology.
  4. The system notifies the primary actor that a new mote is available for configuration in the topology (**Include:** *UC15*: Send notification).

- **Alternative scenarios:** None
- **Remarks:**
  - The mote is pre-configured to connect to a specific gateway by the hardware manufacturer. This linking process is out of scope for this assignment. Likewise, the automatic assignment of an IPv6 address to the mote is out of scope.

3.5 *UC5*: Uninstall mote

- **Summary:** An infrastructure owner deactivates a mote. As a result, the mote no longer sends heartbeats, and the mote and attached pluggable devices are no longer usable.
- **Primary actor:** Infrastructure owner
- **Secondary actors:** Application(s)
- **Interested parties:**
  - *Infrastructure owner:* wants to uninstall a mote and wants the SIoTIP system to correctly reflect the active and available devices.
  - *Customer organisation:* wants their applications to have an accurate view of the active and available devices.
- **Preconditions:**
  - The primary actor has at least one associated mote that was installed previously (cf. *UC4*: Install mote)
- **Postconditions:**
  - The mote and any attached pluggable devices have been marked as 'inactive' in the topology.
  - As a result of the absence of the attached pluggable devices, the applications for which these devices were essential have deactivated themselves (cf. *UC18*: Check and deactivate applications).
- **Main scenario:**
  1. The primary actor physically stops the mote (by unplugging, powering off, or taking it out of the vicinity of the network).
  2. When the system no longer receives heartbeat messages,
     - it marks the mote as 'inactive' in the topology.
     - it looks up the pluggable devices that were attached to the affected mote and marks these as 'inactive' in the topology.
     - it checks for and deactivates applications that are affected by the unavailability of the pluggable devices (**Include:** *UC18*: Check and deactivate applications).
     - It sends a notification to the primary actor (**Include:** *UC15*: Send notification).
- **Alternative scenarios:** None
- **Remarks:** None

3.6 *UC6*: Insert a pluggable device into a mote

- **Summary:** An infrastructure owner plugs in a new pluggable device in one of their motes, which is immediately registered by the system.
- **Primary actor:** Infrastructure owner
- **Secondary actors:** None
- **Interested parties:**
  - *SIoTIP:* wants an extensive infrastructure registered to their platform to support many applications and customer organisations.
  - *Infrastructure owner:* wants an infrastructure capable of supporting the needs of customer organisations, and wants to be able to overview and manage this infrastructure easily.
  - *Customer organisation:* wants to use the infrastructure to run applications to automate processes and increase business value.
  - *MicroPnP hardware manufacturer:* wants to sell as many pluggable devices as possible.
- **Preconditions:**
  - The mote into which the pluggable device is plugged, has been installed previously (cf. *UC4*: Install mote).
- **Postconditions:**
  - The new pluggable device is associated with the primary actor.
  - The new pluggable device can now be initialised in the system for use (cf. *UC8*: Initialise a pluggable device)
- **Main scenario:**
  1. The primary actor plugs the pluggable device into a mote.
  2. The system receives a message from the mote, informing it of the new pluggable device. This message specifies the identifier and type of the new pluggable device.
  3. The system registers the pluggable device as 'uninitialised' and links it with the mote in the topology (cf. *UC10*: Consult and configure the topology).
  4. The system sends a notification to the primary actor (i.e., the infrastructure owner associated to the mote in which the device was inserted) that the new pluggable was detected (**Include:** *UC15*: Send notification).
- **Alternative scenarios:** None
- **Remarks:**
  - Note that the mote does not become available for use by customer organisations and their applications right away. This is only possible after initialising the pluggable for use (cf. *UC8*: Initialise a pluggable device)

3.7 *UC7*: Remove a pluggable device from its mote

- **Summary:** An infrastructure owner removes a pluggable from its mote. This removal is registered immediately by the system. As a result, it is no longer usable.
- **Primary actor:** Infrastructure owner
- **Secondary actors:** Application(s)
- **Interested parties:**

  - *Infrastructure owner:* wants to remove a pluggable device and wants the SIoTIP system to correctly reflect the active and available devices.
  - *Customer organisation:* wants their applications have an accurate view of the active and available devices.

- **Preconditions:**
  - The pluggable device to be removed was previously added by the primary actor (cf. *UC6*: Insert a pluggable device into a mote).

- **Postconditions:**
  - The pluggable device has been marked as 'inactive' in the system.
  - As a result of physically removing the pluggable device, the applications for which the pluggable device was essential have deactivated themselves (cf. *UC18*: Check and deactivate applications).

- **Main scenario:**
  1. The primary actor unplugs the pluggable device from its mote.
  2. A message is sent from the mote from which the pluggable device was removed to the system, containing the identifier of the removed pluggable device.
  3. The system
     - marks the pluggable device as 'inactive',
     - updates the topology,
     - it checks and deactivates applications for which the pluggable device was essential (**Include:** *UC18*: Check and deactivate applications).

- **Alternative scenarios:** None

- **Remarks:** None

3.8 *UC8*: Initialise a pluggable device

- **Summary:** An infrastructure owner initialises a new pluggable device via the SIoTIP Online Service, in order for it to be used by customer organisations and their applications.

- **Primary actor:** Infrastructure owner

- **Secondary actors:** None

- **Interested parties:**
  - *SIoTIP:* wants an extensive infrastructure registered to their platform to support many applications and customer organisations.
  - *Infrastructure owner:* wants an infrastructure capable of supporting the needs of customer organisations, and wants to be able to overview and manage this infrastructure easily.
  - *Customer organisation:* wants to use the infrastructure to run applications to automate processes and increase business value.

- **Preconditions:**
  - The primary actor is logged in (cf. *UC28*: Log in).
  - The primary actor has inserted a pluggable device (cf. *UC6*: Insert a pluggable device into a mote).

- **Postconditions:**
  - The new pluggable device has been marked as 'active' in the topology.

  - The new pluggable device can be used by applications of the customer organisations indicated by the primary actor.

- **Main scenario:**

  1. The primary actor indicates that they want to initialise the newly attached pluggable for use.
  2. The system allows the primary actor to configure the pluggable device in the topology (**Include:** *UC10*: Consult and configure the topology), e.g. to set its physical location.
  3. The system marks the pluggable device as 'active', and informs the primary actor that the pluggable is ready for use.
  4. The system allows the primary actor to configure the access rights for the pluggable device (**Include:** *UC9*: Configure pluggable device access rights).

- **Alternative scenarios:** None

- **Remarks:**

  - Initializing a new pluggable device may result in 'inactive' applications being able to activate (cf. *UC17*: Activate an application).

3.9 *UC9*: Configure pluggable device access rights

- **Summary:** An infrastructure owner configures which customer organisations have access to a specific pluggable device.

- **Primary actor:** Infrastructure owner

- **Secondary actors:** None

- **Interested parties:**

  - *Infrastructure owner:* wants to manage which customer organisations are allowed to use part of the infrastructure they manage.
  - *Customer organisation:* wants to use the infrastructure to run applications to automate processes and increase business value.

- **Preconditions:**

  - The primary actor is logged in (cf. *UC28*: Log in).

- **Postconditions:**

  - The pluggable device can (only) be used by applications of the customer organisations indicated by the primary actor.
  - If the access right for a customer organization was removed, applications for which that access right was essential have been deactivated. (cf. *UC18*: Check and deactivate applications).
  - If an access right for a customer organization was added, applications that could not be activated previously (due to this missing access right) have been activated (cf. *UC17*: Activate an application).

- **Main scenario:**

  1. The primary actor indicates that they want to configure the access rights to pluggable devices.
  2. The system retrieves the list of pluggable devices associated with the primary actor and presents these to the primary actor.
  3. The primary actor indicates for which pluggable device they want to configure the access rights.

4. The system retrieves all customer organisations associated with the primary actor and presents these to the primary actor, thereby indicating which of these customer organizations already have access to the pluggable device.
5. The system asks to indicate which of these should have access to the pluggable device.
6. The primary actor selects the customer organisations that may use the pluggable device and submits the selection.
7. The system
   - updates the profiles of the selected customer organisations, giving them access rights to the pluggable device.
   - checks and activates 'inactive' applications for the customer organisations with updated access rights (**Include:** *UC17*: Activate an application).
   - checks for applications that require deactivation because of the unavailability of pluggable devices (**Include:** *UC18*: Check and deactivate applications).

- **Alternative scenarios:**

  2a. If there are no pluggable devices associated with the primary actor, the system informs the primary actor of this. The use case ends.

  4a. If there are no customer organisations associated with the primary actor, the system informs the primary actor of this. The use case ends.

- **Remarks:** None

3.10 *UC10*: Consult and configure the topology

- **Summary:** An infrastructure owner updates the topology of the infrastructure they manage.
- **Primary actor:** Infrastructure owner
- **Secondary actors:** None
- **Interested parties:**
  - *SIoTIP:* wants an extensive infrastructure registered to their platform, which can be used accurately and effectively by applications and customer organisations.
  - *Infrastructure owner:* wants an infrastructure capable of supporting the needs of customer organisations, and wants to be able to overview and manage this infrastructure easily.
  - *Customer organisation:* wants their applications to have an accurate view of the physical infrastructure, such that their applications can help them increase business value to a maximum extent.
- **Preconditions:**
  - The primary actor is logged in (cf. *UC28*: Log in).
- **Postconditions:**
  - The primary actor has received an overview of the topology, as well as the status of the included hardware devices.
  - The topology is updated according to changes made by the primary actor. This new topology is used from now on.
  - The new topology may result in 'inactive' applications being able to activate (cf. *UC17*: Activate an application).
  - The new topology may result in some applications that require it to have deactivated themselves (cf. *UC18*: Check and deactivate applications).
- **Main scenario:**

1. The primary actor indicates that they want to manage the topology.
2. The system presents the topology for the primary actor.
   - The topology shows a representation of the gateways, and the motes and pluggable devices associated with each gateway, and specifies key relationships between these devices, for example physical location information, or whether pluggable device X and pluggable device Y are interchangeable.
   - New motes and pluggable devices that are not yet included are presented on the side.
3. The primary actor updates the topology.
   - This could involve changing the location of or relations between motes and pluggable devices, or extending the topology with motes and pluggable devices that were not yet included.
4. The primary actor indicates that they are done updating the topology.
5. The system
   - stores the reconfigured topology and informs the primary actor that the reconfigured topology is now being used.
   - activates 'inactive' applications (**Include:** *UC17*: Activate an application).
   - checks for applications that require deactivation (**Include:** *UC18*: Check and deactivate applications).

- **Alternative scenarios:**

  3a. If the primary actor wants to view status details for a specific gateway/mote/pluggable device:
  1. The system composes a detailed overview of the status of the selected gateway/mote/pluggable device (e.g., indicating that the device was unavailable between 2 and 3 a.m.) and presents it.
  2. The primary actor indicates that he/she is done consulting the detailed overview.
  3. Continue from step 2.

  3b. If the primary actor does not want to make changes or consult detailed information, the use case ends.

  5a. If there are still unplaced motes and/or pluggable devices, the system informs that these are not yet usable, but that the newly-reconfigured topology is now in use.

- **Remarks:** None

3.11 *UC11*: Send pluggable device data

- **Summary:** A pluggable device sends data to the system. This could be a sensor (e.g., temperature sensor) sending a reading, or an actuator (e.g., buzzer) sending a status.
- **Primary actor:** Pluggable Device
- **Secondary actors:** Application
- **Interested parties:**
  - *Infrastructure owner:* wants devices in their infrastructure to communicate their data correctly
  - *Customer organisation:* wants applications to effectively receive data in which they are interested.
- **Preconditions:**
  - The pluggable device has been inserted (cf. *UC6*: Insert a pluggable device into a mote).
- **Postconditions:**

  - If the primary actor was initialised for use with SIoTIP (cf. *UC8*: Initialise a pluggable device), the data is stored in the system and forwarded to applications that require it.

- **Main scenario:**

  1. The pluggable device sends data to the system (for example, a light sensor sends its periodic update of the current light intensity level).
  2. The system checks whether the pluggable device has been initialised (cf. *UC8*: Initialise a pluggable device).
  3. If the pluggable device has been initialised, the system
     - stores the data
     - looks up the list of applications that use the pluggable device and for all of these, checks if the data should be forwarded to the application. If this is the case, the system relays the data to the application. (Note: synchronously forwarding all incoming data is not required for all applications, e.g. if an application only wants to receive a reading every 2 seconds, while the pluggable device sends data every second. Also see *UC13*: Configure pluggable device.)

- **Alternative scenarios:**

  3a. If the pluggable device is still uninitialised (cf. *UC8*: Initialise a pluggable device), the sent data is ignored and the use case ends.

- **Remarks:** None

3.12 *UC12*: Perform actuation command

- **Summary:** An application makes one or more pluggable devices (actuators) perform an actuation command.
- **Primary actor:** Application
- **Secondary actors:** Pluggable Device
- **Interested parties:**
  - *Application:* wants to ensure its functionality by triggering actuation commands.
  - *Infrastructure owner:* wants devices in their infrastructure to offer adequate services to applications and customer organisations.
- **Preconditions:**
  - The involved pluggable devices have been initialised for use by the primary actor (cf. *UC8*: Initialise a pluggable device).
- **Postconditions:**
  - An actuation command message was sent to one or more pluggable devices, and the actuation command was performed accordingly.
- **Main scenario:**

  1. An application indicates that it wants one or more pluggable devices to perform an actuation command (e.g., sound all buzzers on the first floor).
  2. The system
     - constructs the actuation command message according to the specific formatting syntax for the involved pluggable device(s)
     - sends the command message to the intended pluggable device(s).

3. The pluggable device(s) receive(s) the actuation command message and perform(s) the contained actuation command.

- **Alternative scenarios:** None
- **Remarks:**
  - The topology may be used to determine the set of actuators that need to perform the actuation command.
  - Note that an explicit acknowledgement is not sent by the pluggable device. However, the pluggable device will regularly send its data, including state information, to the system (cf. *UC11*: Send pluggable device data). This state will reflect the outcome of the actuation command (e.g., indicating that a buzzer is on). Furthermore, the application will be notified when pluggable devices fail (cf. *UC14*: Send heartbeat).

3.13 *UC13*: Configure pluggable device

- **Summary:** An application configures a pluggable device. For example, it adjusts the frequency at which it wants to receive updates of the temperature in a specific room.
- **Primary actor:** Application
- **Secondary actors:** Pluggable Device
- **Interested parties:**
  - *Infrastructure owner:* wants devices in their infrastructure to offer adequate services to applications and customer organisations.
  - *Customer organisation:* wants their applications to easily and accurately use pluggable devices.
- **Preconditions:** None
- **Postconditions:**
  - The application will receive data from the pluggable devices according to the new configuration (cf. *UC11*: Send pluggable device data).
- **Main scenario:**
  1. The primary actor specifies that it wants to set a configuration parameter of a pluggable device.
  2. The system verifies that the value of the configuration parameter is valid for the device (for example, a sensor which provides temperature information may have hardware limits on the sampling frequency).
  3. The system determines whether the pluggable device needs to be reconfigured, and if so, constructs a reconfiguration command according to the specific formatting syntax for the pluggable device and sends it to the pluggable device.
  4. The system updates the internal configuration of the pluggable device.
  5. The system informs the primary actor that the reconfiguration was done successfully.
- **Alternative scenarios:**

  2a. If the value is invalid for the pluggable device, the system informs the application via an exception. The use case ends.
- **Remarks:**
  - Note that different applications may have different preferences for a single pluggable device.

3.14 *UC14*: Send heartbeat

- **Summary:** Each mote periodically sends a heartbeat to the system, informing the system that it and the attached pluggable devices are still operational.
- **Primary actor:** Mote
- **Secondary actors:** None
- **Interested parties:**
  - *Infrastructure owner:* wants to be informed in case of failures in the infrastructure.
- **Preconditions:**
  - The mote was previously installed by an infrastructure owner (cf. *UC4*: Install mote), and, possibly, pluggable devices were attached to it (cf. *UC6*: Insert a pluggable device into a mote).
- **Postconditions:**
  - The system has processed the heartbeat and has reset the timer waiting for the next heartbeat.
  - If pluggable devices are missing from the heartbeat, applications that cannot work without them have deactivated themselves (cf. *UC18*: Check and deactivate applications).
- **Main scenario:**
  1. The system receives a heartbeat from an associated mote. The heartbeat includes a list of the pluggable devices currently attached to the mote.
  2. The system resets the heartbeat timer for the mote which sent the heartbeat, and for all the specified pluggable devices.
  3. The system compares the new list of pluggable devices with the previously known devices for that mote.
  4. If the system detects that pluggable devices are missing, the system may decide that they have failed (e.g., after the device is missing in 3 subsequent heartbeat reports). In this case, the system will deactivate applications for which the affected pluggable devices were essential (**Include:** *UC18*: Check and deactivate applications).
- **Alternative scenarios:**
  - 1a. If the timer for a mote or pluggable expires, the devices are handled as failed devices. Continue from step 4.
- **Remarks:** Appendix A presents technical specifications of the microPnP heartbeat operations.

3.15 *UC15*: Send notification

- **Summary:** A system component was triggered to send a notification to a registered user.
- **Primary actor:** System
- **Secondary actors:** Third-party notification delivery service
- **Interested parties:**
  - *Customer organisation:* wants to monitor the functioning of their applications.
  - *Infrastructure owner:* wants to monitor the physical infrastructure of gateways, motes, and pluggable devices.
  - *Application provider:* wants to be informed of the approval status of their applications and potential problems in newly-uploaded applications.

  - *SysAdmin:* wants to monitor important events that could affect SIoTIP and their customer base.

- **Preconditions:**
  - The system has determined that a notification should be sent to the primary actor. This was triggered by a system component (e.g., because of *UC6*: Insert a pluggable device into a mote, *UC17*: Activate an application, etc.).
- **Postconditions:**
  - A notification describing the occurred event is stored by the system for later consultation (cf. *UC16*: Consult notification message)
  - A notification describing the occurred event was sent to the primary actor via his/her/their registered communication channels (i.e. SMS and/or e-mail).
- **Main scenario:**
  1. The system stores the notification for later consultation (cf. *UC16*: Consult notification message).
  2. The system looks up via which communication channels (i.e. SMS and/or e-mail) the primary actor must be notified.
  3. The system sends the detailed notification describing the occurred event to the primary actor via the communication channel determined in the previous step.
  4. The system marks the notification as 'sent'.
  5. For the delivery channels that support delivery notification (e.g. SMS):
     5.1. The third-party notification delivery service sends back an acknowledgement to the system.
     5.2. The system marks the notification as 'delivered'.
- **Alternative scenarios:**
  5a. If none of the delivery channels supports delivery notification, the use case ends.
- **Remarks:**
  - The delivery method is chosen when the primary actor registered (e.g., *UC1*: Register a customer organisation, or out-of-band for other primary actors) and can be changed later on.

3.16 *UC16*: Consult notification message

- **Summary:** A registered user consults a notification message.
- **Primary actor:** Registered user
- **Secondary actors:** None
- **Interested parties:**
  - *Customer organisation:* wants to monitor the functioning of their applications.
  - *Infrastructure owner:* wants to monitor the physical infrastructure of gateways, motes, and pluggable devices.
  - *Application provider:* wants to be informed of the approval status of their applications and potential problems in newly-uploaded applications.
  - *SysAdmin:* wants to monitor important events that could affect SIoTIP and their customer base.
- **Preconditions:**
  - The primary actor is logged in (cf. *UC28*: Log in).

- **Postconditions:**
  - The primary actor has consulted a notification sent to him/her/them.
  - The consulted notification has been marked as 'read'.
- **Main scenario:**
  1. The primary actor indicates that he/she/they wants to consult a notification.
  2. The system retrieves all notifications for the primary actor and presents them to the primary actor, e.g. as a table or list.
  3. The primary actor selects a notification.
  4. The system presents the details for the selected notification to the primary actor. This includes:
     - A description of the event that occurred.
     - Date and time the event occurred.
     - What triggered the event (e.g., the cause of an application that became inactive)
     - The status of the notification (e.g. 'sent', 'delivered', 'read')
     - The communication channel by which the notification was sent
  5. The system marks the notification as 'read'.
- **Alternative scenarios:**

  2a. If there are no notifications for the primary actor, the systems informs him/her/them of this. The use case ends.
- **Remarks:** None

3.17 *UC17*: Activate an application

- **Summary:** The system activates an application for a subscribed customer organization, resulting in the application being executed in SIoTIP.
- **Primary actor:** System
- **Secondary actors:** None
- **Interested parties:**
  - *Customer organisation:* wants to use applications via the SIoTIP platform.
- **Preconditions:**
  - An application needs to be activated for a customer organization, for example because of a new subscription (cf. *UC19*: Subscribe to application), a change of the topology (cf. *UC10*: Consult and configure the topology), or a new version of the application that automatically replaces earlier versions (cf. *UC22*: Upload an application).
- **Postconditions:**
  - The application is running in SIoTIP.
  - The customer organisation that is subscribed to the application is billed (cf. *UC21*: Send invoice).
  - The involved end-users have access to the application.
  - The application will receive data from the pluggable devices it requires (cf. *UC11*: Send pluggable device data), as configured by the customer organisation to whom the application belongs.
- **Main scenario:**

1. The system checks that all mandatory roles have been assigned to end-users.
2. If all mandatory roles have been assigned, the system checks that all necessary pluggable devices are available in the topology.
3. If all necessary pluggable devices are available, the system activates all necessary parts of the application on gateways and in the Online Service.
4. The system marks the application as 'active' and updates the billing information.
5. The system sends a notification to the customer organisation subscribed to the application, to inform them that the application is running (**Include:** *UC15*: Send notification).
6. The system sends an SMS or e-mail to the end-users that were assigned roles. Possibly, this contains instructions for the end-user on how to install a mobile app linked to the application.

- **Alternative scenarios:**

  2a. If end-users are not assigned to each mandatory role, the application is added as an 'inactive' application and the subscription info is set accordingly. The system notifies the customer organisation subscribed to the application (**Include:** *UC15*: Send notification). The use case ends.

  3a. If not all pluggable devices necessary for the application are available, the application is added as an 'inactive' application and the subscription info is set accordingly. The system notifies the customer organisation subscribed to the application (**Include:** *UC15*: Send notification). The use case ends.

- **Remarks:**

  – In the alternative scenarios, if the activation failed and resulted in an 'inactive' application, the system can still activate the application later on (e.g., when pluggable devices become available or mandatory roles are assigned).

3.18 *UC18*: Check and deactivate applications

- **Summary:** The system deactivates any application that requires deactivation, because of unavailability of essential pluggable devices or unassigned mandatory roles.
- **Primary actor:** System
- **Secondary actors:** None
- **Interested parties:**

  – *Customer organisation:* wants their applications to run correctly and want to be informed in case of problems.

- **Preconditions:** None
- **Postconditions:**

  – Applications of customer organizations for which not all mandatory roles are assigned to end-users, or for which not all essential pluggable devices are available, have been made 'inactive' in the system. Their subscription information is updated accordingly, and the customer organisations have been notified (cf. *UC15*: Send notification).

- **Main scenario:**

  1. The system is triggered to check for applications that require deactivation and deactivate them.
  2. The system composes a list of applications for which either of the following two conditions are not met:
     – All mandatory roles have been assigned to end-users.

  - All essential pluggable devices (i.e., those that are required for the application to function) are available for use and accessible by the application.

3. For each application in the list of applications that do not meet all requirements:
   - The system marks the application as 'inactive'.
   - The system updates the subscription information.
   - The system notifies the customer organisation subscribed to the application (**Include:** *UC15*: Send notification).

- **Alternative scenarios:** None
- **Remarks:**
  - An 'inactive' application can still be re-activated (cf. *UC17*: Activate an application) later on, for example when pluggable devices become available, access rights are configured, or mandatory roles are assigned.

3.19 *UC19*: Subscribe to application

- **Summary:** A customer organisation subscribes to an application via the dashboard.
- **Primary actor:** Customer organisation
- **Secondary actors:** None
- **Interested parties:**
  - *Customer organisation:* wants to use a specific application via the SIoTIP platform.
- **Preconditions:**
  - The primary actor is logged in (cf. *UC28*: Log in).
- **Postconditions:**
  - The system has activated (or has tried to activate) the application for the primary actor (cf. *UC17*: Activate an application).
  - If the newly subscribed application is a newer version of an earlier subscription (cf. *UC22*: Upload an application), the primary actor has been unsubscribed from that earlier version.
- **Main scenario:**
  1. The primary actor indicates they want to subscribe to an application.
  2. The system looks up the available applications (i.e., those to which the primary actor is not yet subscribed, or applications for which a newer version is available that requires a new subscription).
  3. The system presents the available applications to the primary actor, e.g. as a list or table, thereby indicating whether it is a new application or an update.
  4. The primary actor selects the desired application.
  5. The system checks which topology configuration and selection of pluggable devices for the application are necessary (e.g., to indicate in which rooms a heating control application should run) and presents a topology allowing the primary actor to indicate their configuration.
  6. The primary actor carries out the topology configuration.
  7. The system loads the (mandatory and optional) roles to which end-users have to be (or can be) assigned for that application.
  8. The system loads the end-users associated with the primary actor, presents them to the primary actor, and requests to assign end-users to the roles in the application.
  9. The primary actor provides end-users for the roles.

10. The system asks the primary actor if this application should be considered a critical application (cf. *P2*: Requests to the pluggable data database).
11. The primary actor indicates the criticality of the application.
12. The system registers the subscription and criticality of the application.
13. If the selected application is a newer version of an application to which the primary actor was previously subscribed, the primary actor is automatically unsubscribed from that earlier version (cf. step 4 or *UC20*: Unsubscribe from application).
14. The system activates the application (**Include:** *UC17*: Activate an application).

- **Alternative scenarios:**

3a. If there are no applications to which the primary actor can subscribe, the system informs the primary actor. The use case ends without changes.

8a. If no end-users have to be assigned to roles in the application, the use case continues at step 14.

8b. If no end-users are associated with the primary actor yet, the system allows the primary actor to register them (**Include:** *UC2*: Register an end-user). Continue from step 8.

- **Remarks:**
  - This use case concerns both subscribing to a new application, as well as upgrading an existing application to a newer version (if this is not an automatic update, as indicated in *UC22*: Upload an application).
  - Roles in an application can be used to indicate which end-user(s) should be notified in case of specific events.
  - It is possible to reassign end-users to roles later on. For simplicity, no separate use case is provided for this.

3.20 *UC20*: Unsubscribe from application

- **Summary:** A customer organisation unsubscribes from an application.
- **Primary actor:** Customer organisation
- **Secondary actors:** None
- **Interested parties:**
  - *Customer organisation:* no longer wants to use a specific application.
- **Preconditions:**
  - The primary actor is logged in (cf. *UC28*: Log in).
- **Postconditions:**
  - The primary actor is unsubscribed from the application, and this application is deactivated in the system.
  - An invoice for outstanding fees for that application has been sent to the third-party invoicing service.
- **Main scenario:**

1. The primary actor indicates they want to unsubscribe from an application.
2. The system looks up a list of applications the primary actor is subscribed to, and presents this overview to the primary actor, e.g. as a list or table.
3. The primary actor indicates which application they want to unsubscribe from.

4. The system (i) ends the subscription for the selected application, (ii) deactivates the application, and (iii) constructs an invoice for outstanding payments for that application (cf. *UC21*: Send invoice).
5. The system informs the primary actor that the unsubscription was successful.

- **Alternative scenarios:**

  2a. If the primary actor is not subscribed to any applications, the system informs the primary user. The use case ends.

- **Remarks:** None

3.21 *UC21*: Send invoice

- **Summary:** A customer must be billed, e.g. for a subscription to an application.
- **Primary actor:** Customer organisation
- **Secondary actors:**
  - Third-party invoicing service (e.g. Zoomit or credit card company)
- **Interested parties:**
  - *Customer organisation:* wants to simplify paying its (recurring) fees for using the SIoTIP system.
- **Preconditions:**
  - The system has detected that a new invoice must be sent to the primary actor, e.g. it is the end of the month in which a customer organisation must pay for its application subscriptions.
- **Postconditions:**
  - An invoice for the provided services or purchased products has been sent to the primary actor via the third-party invoicing service.
- **Main scenario:**
  1. The system constructs an invoice containing:
     - the provided service or product (e.g. monthly subscription fee for applications or purchased hardware),
     - the amount to be paid,
     - the date the payment is due and
     - the contact details of the primary actor (e.g. Zoomit identifier or credit card data)
  2. The system sends the invoice to the third-party invoicing service.
- **Alternative scenarios:** None
- **Remarks:**
  - Following up on the payment of these invoices is out of scope for this system.

3.22 *UC22*: Upload an application

- **Summary:** An application provider uploads a new application, or a new version of an existing application.
- **Primary actor:** Application provider
- **Secondary actors:** None
- **Interested parties:**
  - *SIoTIP:* wants as many high-quality applications as possible available on their platform for customers.
  - *Customer organisation:* wants applications that suit their needs of automating processes and increasing business value.
  - *Application provider:* wants to make his/her applications available on SIoTIP to reach a large amount of customers.
- **Preconditions:**
  - The primary actor is logged in (cf. *UC28*: Log in).
- **Postconditions:**
  - If an new application is uploaded, the application is made available in the system. Hence, customer organisations are able to subscribe to it via the application store (cf. *UC19*: Subscribe to application).
  - If an updated version of an existing application is uploaded and the primary actor wants it to automatically replace the previous version, it is deployed and existing subscriptions of that application are updated (cf. *UC17*: Activate an application).
- **Main scenario:**
  1. The primary actor indicates that he/she wants to upload an application.
  2. The system asks the primary actor if he/she wants to update an existing application.
  3. The primary actor provides his/her choice.
  4. If the primary actor indicated that he/she wants to update an existing application, the system composes an overview of applications uploaded by the primary actor (e.g., as a list or table), presents this, and requests the primary actor to indicate which application to update.
  5. The primary actor chooses the application to be updated.
  6. The system asks the primary actor if this is an update that should be automatically activated for existing subscriptions.
  7. The primary actor provides his/her choice.
  8. If the primary actor has indicated that existing subscriptions should be automatically updated, the system requests the versions (or range of versions) that should be automatically updated.
  9. The primary actor provides the requested information.
  10. The system asks the primary actor to upload the application code, a description for display to customer organisations, and meta-data (such as the version number and subscription price).
  11. The primary actor uploads the requested information.
  12. The system performs automated application checks.
  13. If the checks were successful, the system sends a notification to the primary actor (**Include:** *UC15*: Send notification).
  14. The system makes the application available in the application store. In case of an updated application, it replaces the previous versions of the application.

15. In case of an updated application that should automatically be activated for existing subscriptions (cf. step 6), the system updates the existing subscriptions of the application to the new version (**Include:** *UC17*: Activate an application).

- **Alternative scenarios:**

4a. If the primary actor has indicated that he/she wants to upload a new application, the use case continues from step 10.

8a. If the primary actor has indicated that existing subscriptions should not be automatically updated, the use case continues from step 10.

13a. If the checks were not successful, the system notifies the primary actor and a SysAdmin (**Include:** *UC15*: Send notification). The use case ends.

- **Remarks:**
  - For updated applications, application providers have the choice between automatically updating all application subscriptions of the previous version, or requiring customer organisations to subscribe to the new version individually (cf. *UC19*: Subscribe to application). The latter could be used if the application provider requires the customer organisations to pay for the new version. They may, however, also use the former model for some updates, such as minor improvements and bugfixes. Requiring the customer organisations to explicitly subscribe to the new version can also be useful in case the new version requires new hardware.
  - Explicitly indicating the versions from which to automatically update (in step 8) is necessary to prevent the automatic (and free) update from v2.0 to v2.1 to also be installed for v1.x subscriptions, thereby bypassing the paid upgrade from v1.x to v2.0.
  - For simplicity, the workflow of a sysadmin that manually reviews and approves/rejects an application that has failed automatic testing (as mentioned in Part 1 of the assignment) is not part of this assignment.

3.23 *UC23*: Consult application statistics

- **Summary:** An application provider consults statistics about the applications he/she uploaded, such as the amount of subscribers and approval information.
- **Primary actor:** Application provider
- **Secondary actors:** None
- **Interested parties:**
  - *Application provider:* wants to use SIoTIP to make his/her applications available for a large customer base, and wants to easily manage these applications.
- **Preconditions:**
  - The primary actor is logged in (cf. *UC28*: Log in).
- **Postconditions:**
  - The primary actor has consulted statistics about the uploaded applications.
- **Main scenario:**

1. The primary actor indicates that he/she wants to an overview of uploaded applications.
2. The system retrieves all applications uploaded by the primary actor, and information about the amount of subscribers for that application. It presents this overview to the primary actor.
3. The primary actor selects an application.
4. If the selected application is an approved application, the system presents detailed statistics including the amount of subscribers.

- **Alternative scenarios:**

  4a. If the selected application is an unapproved application, the system presents detailed information including a description of why the check failed.

- **Remarks:** None

3.24 *UC24*: Consult historical data

- **Summary:** An application wants a historical overview of data collected by SIoTIP, e.g. the temperature in room X in the last 30 minutes.
- **Primary actor:** Application
- **Secondary actors:** None
- **Interested parties:**
  - *Customer organisation:* wants to use their applications to increase business value, which includes getting clear and correct overviews of data.
- **Preconditions:** None
- **Postconditions:**
  - The primary actor has received a historical overview of the requested historical data.
- **Main scenario:**
  1. The primary actor indicates that it wants to consult a specified collection of historical data in a specified timeframe.
  2. The system determines from which pluggable devices the data is required and looks up the data.
  3. The system presents the primary actor with the requested historical overview, e.g. as a table.
- **Alternative scenarios:** None
- **Remarks:** None

3.25 *UC25*: Access topology and available devices

- **Summary:** An application wants a view of the topology and devices it has access to.
- **Primary actor:** Application
- **Secondary actors:** None
- **Interested parties:**
  - *Infrastructure owner:* wants an infrastructure capable of supporting the needs of customer organisations.
  - *Customer organisation:* wants their applications to have an accurate view of the physical infrastructure, such that their applications can help them increase business value to a maximum extent.
- **Preconditions:** None
- **Postconditions:**
  - The application has received an overview of the topology, including the pluggable devices available to it.
- **Main scenario:**

1. The primary actor indicates that it wants an overview of the topology.
2. The system looks up the pluggable devices that are available to the customer organisation that owns the primary actor, and composes a view on the topology including these pluggable devices.
3. The system presents the topology view to the primary actor.

- **Alternative scenarios:** None
- **Remarks:** None

3.26 *UC26*: Send application command or message to external front-end

- **Summary:** An application sends an application command to another part of the same application (e.g., from a part running on the gateway to a part in the Online Service), or a message to a front-end external to SIoTIP (e.g., a mobile app).
- **Primary actor:** Application
- **Secondary actors:** None
- **Interested parties:**
  - *Customer organisation:* wants to use their applications to increase business value, which may include communicating with other applications and systems.
- **Preconditions:** None
- **Postconditions:**
  - The application command or message was sent to the destination application or system, or the primary actor has been informed of the failure thereof.
- **Main scenario:**
  1. The primary actor indicates it wants to send an application command or message to an external front-end and specifies the destination (e.g., as an application identifier for SIoTIP applications, or a hostname and port for external systems).
  2. The system checks that the primary actor is allowed to send to the specified destination.
  3. If the primary actor is allowed to send to the destination, and if the destination is another application, the system delivers the application command to that destination (**Include:** *UC27*: Receive application command or message from external front-end).
  4. The system informs the primary actor that the message was sent.
- **Alternative scenarios:**
  - 3a. If the primary actor is allowed to send to the destination, and if the destination is an external application or system, the system looks up how to reach the destination (e.g. via a TCP connection with the destination) and asynchronously sends the message. Continue from step 4.
  - 3b. If the primary actor is not allowed to send to the destination, the system informs the primary actor of this problem. The use case ends.
- **Remarks:**
  - The destination could be another part of the same application running in SIoTIP (which is received as described in *UC27*: Receive application command or message from external front-end) or a front-end on an external system (e.g. an accompagnying mobile app) which uses the library provided by the SIoTIP Corporation to interface with applications running in SIoTIP).

3.27 *UC27*: Receive application command or message from external front-end

- **Summary:** An application receives an application command from another part of the same application (e.g., a part running in the Online Service from a part on a gateway) or a message from a front-end external to SIoTIP (e.g., a mobile app).
- **Primary actor:** Application
- **Secondary actors:** None
- **Interested parties:**
  - *Customer organisation:* wants to use their applications to increase business value, which may include communicating with other applications and systems.
- **Preconditions:**
  - An application command was sent by another application in SIoTIP (cf. *UC26*: Send application command or message to external front-end) or a message was sent by a front-end on an external system (e.g., a mobile app).
- **Postconditions:**
  - The application command or message was delivered to the recipient application.
- **Main scenario:**
  1. The system receives an application command or message for a SIoTIP application.
  2. The system checks that the destination is available.
  3. If the destination is available, the system delivers the message to the destination application.
- **Alternative scenarios:**

  3a. If the destination is not available (e.g., because it has become inactive), the system sends back a message to inform the sender. The use case ends.
- **Remarks:**
  - The sender could be another application running in SIoTIP (cf. *UC26*: Send application command or message to external front-end) or a front-end on an external system (e.g. an accompagnying mobile app), which uses a library provided by the SIoTIP Corporation to interface with applications running in SIoTIP.

3.28 *UC28*: Log in

- **Summary:** A registered user wants to use the SIoTIP Online Service and logs in by providing authentication credentials.
- **Primary actor:** A registered user
- **Secondary actors:** None
- **Interested parties:**
  - *SIoTIP:* wants to authenticate its users for access control and traceability.
  - *Customer organisation:* wants to manage applications and roles of end-users in applications.
  - *SysAdmin:* wants access to SIoTIP to manage the system.
  - *Application provider:* wants to SIoTIP to manage his/her applications.
  - *Infrastructure owner:* wants access to the system to carry out infrastructure management tasks, and only wants authorised entities to use their infrastructure.

- **Preconditions:**
  - The primary actor is registered (cf. *UC1*: Register a customer organisation for end-users, out-of-band for other types of primary actors).
- **Postconditions:**
  - The primary actor is logged in and can use the SIoTIP Online Service.
- **Main scenario:**
  1. The primary actor indicates he or she want to log in to the system and provides authentication credentials, for example a user name and password.
  2. The system verifies the provided authentication credentials.
  3. If the provided credentials are correct, the system confirms successful login to the primary actor and logs him or her in.
- **Alternative scenarios:**
  - 3a. If the provided credentials are incorrect, the system notifies the primary actor of this. Continue from step 1.
- **Remarks:**
  - This use case abstracts the used method of authentication. For example, authentication can be done using techniques such as username-password, security tokens, cryptographic keys, etc. The employed method depends, among others, on the type of primary actor.
  - Registration of customer organisation and end-users is described in UC1. Registration of infrastructure owners, SysAdmins, and application providers is done out-of-band, and hence not described in use cases.

3.29 *UC29*: Log out

- **Summary:** A logged-in user signs out of the SIoTIP Online Service.
- **Primary actor:** A registered user
- **Secondary actors:** None
- **Interested parties:**
  - *SIoTIP:* wants to authenticate its users for access control and traceability.
  - *Customer organisation:* wants to manage applications and roles of end-users in applications.
  - *SysAdmin:* wants access to SIoTIP to manage the system.
  - *Application provider:* wants to SIoTIP to manage his/her applications.
  - *Infrastructure owner:* wants access to the system to carry out infrastructure management tasks, and only wants authorised entities to use their infrastructure.
- **Preconditions:**
  - The primary actor is logged in (cf. *UC28*: Log in).
- **Postconditions:**
  - The primary actor is logged out.
- **Main scenario:**
  1. The primary actor indicates he or she wants to log out of the system.
  2. The system logs the primary actor out and indicates success to the primary actor.

- **Alternative scenarios:** None
- **Remarks:**
  - This use case abstracts from the method of authentication that is used (cf. *UC28*: Log in).

A MicroPnP interfaces

The interfaces below describe the functionality offered by the MicroPnP platform. They consist of two parts: (a) the interfaces on the gateway that are called by the pluggable devices (section A.1) and, conversely, (b) the interfaces of the pluggable devices that are called by the gateway (section A.2). The datatypes that are used in these APIs are described in section A.3.

Note: *The operations in these interfaces do not raise any exceptions, so sending incorrect actuation or configuration commands to a pluggable has no effect.*

A.1 Gateway – Provided interfaces

- Heartbeat
  - `void heartbeat(MoteInfo moteInfo, List<Tuple<PluggableDeviceID, PluggableDeviceType>> pds)`
    - Effect: Periodic heartbeat from the mote to the gateway, including a list of the `PluggableDevice`s and their `DeviceType`s (i.e. those currently plugged into the mote)
    - Exceptions: None
- DeviceData
  - `void rcvData(PluggableDeviceID pID, DeviceData data)`
    - Effect: Provide pluggable device data to the gateway *(Initiated by the device)*.
    - Exceptions: None
  - `void rcvDataCallback(PluggableDeviceID pID, DeviceData data, int requestID)`
    - Effect: Provide device data to the gateway *(Callback of getDataAsync)*.
    - Exceptions: None
- DeviceManagement
  - `void pluggableDevicePluggedIn(MoteInfo mInfo,PluggableDeviceID pID, PluggableDeviceType type)`
    - Effect: Notify the gateway that a new `PluggableDevice` of the given type is connected to the mote.
    - Exceptions: None
  - `void pluggableDeviceRemoved(PluggableDeviceID pID)`
    - Effect: Notify the gateway that a `PluggableDevice` is removed.
    - Exceptions: None

A.2 PluggableDevice – Provided interfaces

- Config
  - `Map<String,String> getConfig()`
    - Effect: Returns the current configuration of a `PluggableDevice` as a parameter-value map.
    - Exceptions: None.
  - `boolean setConfig(Map<String,String> config)`
    - Effect: Set the given configuration parameters of the `PluggableDevice` to the given values. Setting unknown parameters on a `PluggableDevice` (e.g., 'noise threshold' → '3' on a light sensor) has no effect.
    - Exceptions: None.
- RequestData

  - `DeviceData getData()`
    - Effect: Synchronously retrieve the device data of a device.
    - Exceptions: None.
  - `void getDataAsync(int requestID)`
    - Effect: Asynchronously retrieve the device data of a device *(by calling rcvDataCallback)*.
    - Exceptions: None.

- Actuate
  - `void sendActuationCommand(String commandName)`
    - Effect: Send an actuation command to the actuator. Sending an unknown actuation command has no effect.
    - Exceptions: None.

A.3 Datatypes

- `MoteInfo`: A object containing information on a mote. This is a list of key-value pairs. The values depend on the type of mote. For example, only a battery-powered mote would include the batterylevel info.
  - `int moteID`
  - `int manufacturerID`
  - `int productID`
  - `int batteryLevel` *(Only for battery-powered motes.)*
- `PluggableDeviceID`: A unique identifier of the `PluggableDevice`.
- `PluggableDeviceType`: Denotes the type of the pluggable device.
- `DeviceData`: Data from a pluggable device. For sensors, this contains sensor values. For actuators, this contains the state of the actuator. The data is encapsulated within a JSON message, and should be converted into something meaningful based on the device type of the pluggable device that sent the data.
- `Map<String,String>`: A map of key-value pairs. E.g., 'sensitivity' → '10'.