Chapter 11

Toward a Concrete Architecture

Christine Jardak and Joachim W. Walewski

11.1 Objective and Scope

This Section serves to illustrate how the IoT ARM can be used for the generation of concrete architectures. This goal is pursued by applying the IoT ARM to a concrete use case and application scenario. This Section serves thus as a complement to Sect. 6.3. Notice that we are not providing all the details that would usually be part of an architecture description, rather, the idea is to illustrate aspects of the architecture actions elaborated on in Sect. 6.3.

Throughout this Section we provide summaries of how the description provided here illustrates statements made elsewhere in the document, for instance Sect. 6.3. In such summaries we occasionally also discuss how complementary actions to those laid out in Sect. 6.3 can enhance the architecting process. All such meta-commentary is set apart in light-grey boxes like this one.

The targeted use case of this architecture is a combination of *Pay-By-License-plate* (PBL) parking and *Recognise-By-License-plate* (RBL) parking enforcement. The core idea of such a system is to use the license plate of a car as a unique identifier for on-street parking. Upon purchase of a time-parking permit, the customer provides the license-plate number of her car for identification. This parking feature shall be available to time parkers and residents. Examples for time parkers are tourists, and locals from a suburb who visit the city centre for shopping, restaurant visits, etc. Residents are defined as denizens of a municipality,

C. Jardak (✉) • J.W. Walewski
Siemens AG, Otto-Hahn-Ring 6, 81739 Munich, Germany
e-mail: christine.jardak@siemens.com; joachim.walewski@siemens.com

and they are purchasing a parking permit for an extended period of time for on-street parking in the vicinity of their residence. By using the license-plate number as ID for the parked car, paper copies of the parking permit do not longer have to be placed on the dash board of the parked cars. In such a system, the license plate is also used by the parking enforcement for checking the permit of the car against a data base provided by the parking service itself. More information on PBL and RBL can be found elsewhere in the literature (Digital Payment Technologies 2013; Genetec 2013). In the remainder of this Section we refer to this envisaged system as a PBL system.

It should be also noted that the entire system is to be designed in a way that it can be made part of a version update of an already existing central system that manages municipal on-street parking lots.

Notice that scopes usually are part of the business goals. Depending on the complexity of the use case such description can be rather complex and long. Besides describing the goal of the system, the description also needs to include a sketch of how one intends to achieve this goal. Without a spelled-out approach, it is impossible to generate an architecture.

Also notice that due to resource and time constriction we were not able to dedicate the same level of attention to all the steps in the architecting process as laid out in Sect. 6.3. In particular, no Functional Decomposition, Interactions, nor interface definitions are provided. Also, neither the Deployment nor the Operational Views are touched upon.

11.2 Physical Entity View and IoT Context View

11.2.1 *Physical Entity View*

This Section relates to Sect. 6.3 and Chap. 6 Figure 3. In the referenced Section, the content and the importance of the Physical-Entity View are discussed. Here, we provide a concrete example of the PE View for the PBL system presented in the previous Section. Notice that this view can be much more complex for other use cases. For instance, if the state of the Physical Entity is going to be inferred from a wide range of measured physical quantities, one not only needs to catalogue these quantities (viewpoints!), but also their range and how these ranges translate into the

(continued)

(continued)

qualitative states that are to be inferred from the measured quantities. An illustrative use case for this is the Red-Thread example (see Sect. 4.2), viz. the transport of orchids. One needs a rather fine-tuned model of the orchids in order to infer their current condition from environmental quantities such as air temperature and humidity and the duration for which the orchids have been exposed to these conditions.

As briefly described in Sect. 11.1, the thing at the core of the IoT system is the car. More specifically, the entity of interest is the **parked** car. Therefore, the Physical Entity in the IoT Domain Model (see Sect. 7.3) is the parked car. An example of the Physical Entity is shown in Fig. 11.1.

Notice that the parking lot itself is not the Physical Entity but the car. That this is the case is not an intrinsic property of the Physical Entity, rather of what the business goals behind the envisaged architecture are, and how they will be achieved (the aforementioned approach).

As described in Sect. 11.1, the goal of the envisaged IoT system is to implement one service for both time and resident parkers, and the car's license plate was chosen upfront as the unique identifier for both use cases. The parking lot becomes an entity of interest when, for instance, the parking enforcement enquires whether a parked car is authorised to park at that specific location. However, since this is only one of the envisaged use-case scenarios (see below) where the parking lot could qualify as the Physical Entity, the parked car and not the parking lot is chosen. This does not imply that there can only be one Physical Entity per IoT system. Rather, one Physical Entity type turns out to be sufficient in order to meet the system goal as described in Sect. 11.1.

Notice that the process provided in Chap. 6 indicates that the Physical View is contingent on the business goals: once the goals are chosen the Physical Entity can be identified together with the properties about the Physical Entities that are of interest for the IoT system. This dependency is illustrated in the above example.

Notice that since only the license plate is used to identify the parked car, the envisaged system can readily encompass the parking of motor bicycles and the like. What is paramount is that it is a vehicle that is identifiable through its license plate.

Fig. 11.1 A car parked in the Gatwick North Terminal Flightpath long stay car park (Whittington 2010)

As shown in Chap. 6 Figure 3, both the Physical View and the business goals inform the IoT Context View. In the next section we illustrate this inter-relatedness for the PBL architecture.

11.2.2 *IoT Context View*

As already stated in Sect. 11.1, the envisaged system is to be integrated with an existing system for the control of parking-payment systems, which we refer to as Control Centre. In other words, the system envisaged is an extended version of the existing system. Future extensions are very likely.

The context diagram of the PBL system is shown in Fig. 11.2.

As described in Sect. 6.3.2, the context view describes “the relationships, dependencies, and interactions between the system and its environment” (Rozanski and Woods 2011). While we describe some inner structure of the envisaged system, viz. an enhanced version of the Control Centre, this level of detail is not mandatory. What is mandatory though is to provide an information about (Rozanski 2013).

- System scope and responsibilities
- Identity of external entities and services and data used
- Nature and characteristics of external entities
- Identity and responsibilities of external interfaces
- [Nature and characteristics of external interfaces]
- Other external interdependencies

(continued)

Fig. 11.2 Context diagram of the PBL IoT system. The dashed box indicates the border of the Control Centre (© comeo.de)

- (continued)
- [Impact of the system on its environment]
 - [Overall completeness, consistency, and coherence]

Note that for the sake of brevity of this section we will ignore the aspects between brackets.

The nature of external interfaces was not addressed since this information was not available at the time of writing. Besides the mission statements in Sect. 11.1 we cannot yet predict how this system impacts its environments. This question can often only be addressed when the system is implemented and tested. The overall completeness, consistency, and coherence of the system was not addressed here, since, due the simplicity of the use case, and the strong boundaries put onto it by the business model (for instance, off-street parking is excluded), we felt that this item is fulfilled by default. Also notice that system scope already was provided in Section 0 In a regular architecture description, the scope is part of the business goals. There is thus a natural overlap of context view and business goals. In case the business goal already contains the full information about system scope and responsibilities,

(continued)

Table 11.1 Types of parkers and the services to be offered

Type	Description	Services to be offered
Resident parker	Lives in the vicinity of the parking lot used. Needs to park on a frequent but not necessarily on a daily basis. Purchases a subscription for this type of parking	PBL on a subscription basis. Subscription shall be possible via walk in at the local Registry Office. Other access modalities include mail, email, web services, and calls
Time parker	Needs to park for a limited time interval on a location that is typically not in the vicinity of the driver’s residence. Envisaged usages encompass short-time city parking (for instance for shopping) but also extended-stay parking at, for instance, airports	PBL on a pay-by-need basis. This type of permits shall be purchasable at PDMs, but also through web services (for instance, a smart-phone application)

(continued)

this information does of course not need to be repeated in the context view, but can rather be cross-referenced. Notice that the context view can be kept rather descriptive, but this cannot be done at the expense of completeness.

The Control Centre, which is the focus of the IoT architecture to be devised, is seen at the centre of Fig. 11.2. Also shown are purchase/transaction operations by the time parkers and the resident-parkers. The two types of parkers and what services shall be offered to each of them are summarized in Table 11.1. Figure 11.2 also contains on-street *Pay-and-Display Machines* (PDMs), parking enforcement, and the registry office. The latter maintains a database on resident parkers (name, address, permit purchased, etc.). What is inside and outside the scope of the IoT architecture to be generated is summarised in Table 11.2.

Notice that Table 11.1 can alternatively be part of the business goals (description of end customers and the services to be offered to them).

As shown in Chap. 6 Figure 3, the IoT Context View consists of two parts, the context view and the IoT Domain Model. In many cases it will be easier to construct the context view first, since (a) one does not yet need to understand the inner workings of the envisaged IoT system, and (b) the context view

(continued)

Table 11.2 Overview of what components and interfaces in the context diagram (see Fig. 11.2) are part of the architecture to be devised

Within scope of the system architecture	Outside the scope of the system architecture
Control Centre; interfaces to PDMs, parking enforcement, and Registry Office. Parker; car	PDMs; web services for interacting with Control Centre (online time-parking tickets) and with Registry Office (Resident Parking Registration); Enforcement system

(continued)

focuses on the interfaces and what lies outside of the IoT system. The amount of detail on “outside interfaces” and the outside itself is usually much less than that of the IoT system itself.

11.2.2.1 Business Goals Revisited

As already mentioned the envisioned IoT system extends and improves existing car parking system. In the following we provide more information about the actors and devices involved, and also how their functionalities and roles are going to change due to the envisaged system enhancement. Such a detailed discussion is valuable not only from a mission-statement point of view, but also from an IoT-Domain-Model point of view, since it provides valuable additional information about the entities that form the IoT Domain Model, and how these entities interact.

Notice that contrary to the partition prescribed in Chap. 6 Figure 3, business goals and the IoT Context View (and the Physical Entity View) can of course be provided in one contiguous part of the architecture description. Such an aggregate presentation can make sense since all three (as in the example provided here) are characterised by a strong interdependence (chicken-and-egg problem!)). If these two/three descriptions are indeed bunted together this needs of course to be clearly flagged in the table of content of the architecture description.

In this section, we shed more light on the planned improvement of the parking system by comparing the current functionalities of the entities in the context diagram with how they are going to look like after the planned improvement.

Pay-and-Display Machines (PDM)

Today: Parking Ticket Identification

PDMs are mounted on the side of public roads and have a major task of managing on-street parking places (Wikipedia 2013g). They allow a driver to buy a time-limited parking permit for a defined geographic region of on-street parking lots. After paying the parking fee, the PDM prints out the corresponding parking ticket. The driver is tasked to place the parking ticket visibly on the dashboard of her car.

Enhancement: Pay-by-License Plate

Our target is to simplify on-street parking by allowing the driver to head toward the nearest PDM, to type in the license plate number of her car, and to pay the parking fee. In this scenario she does not need to place a printed parking permit on the dashboard of her car. Instead the information entered (license plate) and the information about the permit (begin, end, zone) is communicated from the PDM to the Control Centre (see Fig. 11.2), where it is stored in a Parking-White-List database. The information stored in the database can be accessed by the parking-enforcement authority operating in the pertinent precinct of the municipality (see the below entry on the Registry Office).

Control Center

Today: PDMs Monitoring Centre

The Control Centre is a monitoring centre for multiple PDMs. It supplies PDMs with new parameter data such as current parking fees. It also monitors the PDM transaction data, cash-box status, and it provides statistical data (e.g., # of tickets sold) and status messages about the monitored PDMs to the users of the Control Centre.

A Control Centre is not always managed by the municipality itself. In many cases the management of the system is outsourced to a private company, but the municipality remains the owner of the data.

Notice that ownership of the system parts (Control Centre) and also of the out-of-system parts (Registry Office, parking enforcement ...) has direct implications for the requirements engineering (functional view, information view, security-risk analysis ...).

Enhancement: Connection to Web, and to the Registry Office

Instead of buying a paperless time-parking permit from a PDM, drivers are given the possibility to execute the purchase online. To do so, a driver logs in to the corresponding webpage or installs an app on her smart device. The driver provides her license plate number, the parking zone, the parking time interval, and finalises her purchase by paying parking fee. The aforementioned information is then stored in the Parking-White-List database.

Registry Office*Today: Registering Residents and “Sticky” Permits*

A registry office maintains a municipal database containing information on the current residence of persons and their permits. Residents can register for a parking permit by, for instance, providing pertinent information on the Registry-Office’s Internet website; by sending the information by mail; or even by visiting the Registry Office in person. A standardised permit is handed to registered resident parkers. These permits are usually fixed to the car (glued to the inside of the windshield, etc.).

Enhancement: The municipality offers “immaterial” permits. Here, the license-plate number is used for the identification of cars that are allowed to park on a resident-parking term. Information about these cars, viz. their license-plate numbers, the zones where they are allowed to be parked, and when they are allowed to be parked are provided to the Control Centre, which stores this information in a Parking-White-List database.

Parking-White-List Database

The purpose of this database is to maintain a parking white list, i.e. a list of cars -identified by their license-plates that are permitted to park within the region managed by the Central System. Besides the license-plate number, the white list also provides the geographical region, where the pertinent cars can be parked, and also when parking commences and when it ends.

Today: No Parking-White-List Database

While statistics about, for instance, how many permits have been purchased from the PDMs (see, for instance (Island Group 2012)) can be retrieved from the database, no identification about the parked cars is performed in the current system.

Enhancement: Parking-White-List in Control Centre

A Parking-White-List for time-parkers and resident-parkers is made part of the Control Centre. The content of the Parking-White-List is gathered from three sources: PDMs, web services, and the Registry Office (see Fig. 11.2). The first two sources provide the Parking-White-List with information updates about time-parkers that have booked via PDMs or via the Internet. Information about resident parkers is provided by the Registry Office.

Enforcer/Handheld

Today: Controlling Parking Tickets and Resident Parking Permits

The task of the enforcer is to control whether a car is authorised to be parked in a specific zone and at a specific time. In the most common scenario, the enforcer is equipped with a handheld device that is capable of printing a paper ticket to be left at the vehicle. The handheld ensures that a variety of checks are executed on the data in order to eliminate invalid entries such as misspelled street names. The entered data is then transferred from the handheld either overnight or immediately via, for instance, GPRS to a back-end office, where the information about issued parking-violation tickets is stored.

Enhancement: Controlling the License Plate Number Only

The task of the enforcer changes in a sense that she does not need to struggle with badly visible parking tickets and resident parking permits in order to read and enter the data in the handheld. Instead, she scans the license-plate number of a parked car with her handheld. In the next step, she uses the handheld for checking the license-plate number together with the geographical location of the parked car, against the Parking-White-List. Notice that in this enhanced scenario, neither time-parkers nor resident-parkers need to place any permit visibly in their car.

11.2.3 IoT Domain Model as an Expansion of the Context View

As discussed in Sect. 6.3, in the IoT-A architecting process, the IoT Domain Model is generated in order to enrich the standard context view with IoT-specific context and with more details about the inner workings of the system. The latter is important for, among others, the requirements process

(continued)

Fig. 11.3 IoT Domain Model of the PBL system

(continued)

(see Sect. 6.4). Such a domain model also stipulates entity names and relationships to be used in the requirement process and for the derivation of other architectural views (functional view, information view, deployment view ...).

The IoT Domain Model for the PBL system is shown in Fig. 11.3.

Notice that major input for Fig. 11.3 was derived from the previous Section on business goals.

Next we describe the steps we took for deriving the IoT Domain Model depicted in Fig. 11.3. After that, we discuss the particularities of the entities in the IoT Domain Model.

The previous figure provides an enriched and IoT-specific viewpoint to the context. This relates to the context diagram in Fig. 11.2; The legend reads as follows:

- In Yellow: human users;
- In Green: software;
- In blue: hardware;
- In beige: concept that fits in none of the previous categories.

- Classes with thick boundary lines: part of the architecture description. The architecture also covers all associations originating from or terminating at the Control Centre.
- Dashed boxes: system borders. Notice that only the Control Centre is within the system scope of the generated architecture.

11.2.3.1 Modelling Steps

System Users

The human/institutional system users can usually readily be inferred from the business goals and the context view (see above). In the following text we summarises the available information about the users so that we next can apply the IoT-Domain-Model mapping exercise in Sect. 9.1.7 (see Sect. 9.8).

Generally, the users of a system are interested in its functionalities. Our system has three functional outputs: it introduces a simple parking procedure, which is PBL; it allows to easily identify illegal parkers by means of RBL; and it increases parking revenues and public order due to quick spotting and processing of parking violations.

Who is interested in these functional outputs? By answering this question, we can determine the different categories of system users. In our use-case, the PBL is an interesting functionality for parkers, the RPL for enforcers, and the increase of the parking revenues and public order is obviously of a high interest for the municipality, e.g. the registry office. In the following, we define the users of each category.

- **Parkers:** human users. We distinguish between two types parkers: A resident-parker and a time-parker (see Table 11.1). The first is a resident that would like to have an affordable and easy solution to park his car on the street in his neighbourhood. The second is a driver that needs to park his car on a street for a limited period of time in order to accomplish a local activity. The time parker departs after the local activity is completed.
- **Enforcer:** The enforcer in this case, could be the human that uses the handheld or, being more granular, it could even be the application software that runs on the handheld and which is used by the enforcer. The IoT DM is flexible in terms of the granularity of modelling. We decided to model the enforcer as a user.

- **Registry office:** The Registry Office is a municipal office that can be considered as a system user. For the sake of clarity, we note for the reader that other offices, such as the public-order office and the police can be also be modelled as system users. Here, we only model the Registry Office as a user, since the other entities are not part of the business model. However, in a future extension of the system, these entities could of course come into the scope of the PBL Service and would then be added to the IoT Domain Model. This user provides the system with the newest information of the subscribed cars to the PBL Service.

Notice that the system users can be identified with a similar question as for the Physical Entity. In the latter case one asks the question what physical entity the system needs to interact with in order to fulfill its business goal. In the case of system users one asks who is interested in the output generated from system. This output encompasses of course also information inferred from interacting with the Physical Entity.

Procedure Application

In this section we model the different parts of our system (see Fig. 11.3) by applying the six-step procedure, in Sect. 9.1.7 to each of the four system-users: resident-parker, time-parker, parking enforcer, and the Registry Office. This six-step procedure yields six answers (A1 to A6), which are discussed below.

Resident-Parker

In order for a resident-parker to use the parking PBL facility, he needs to subscribe to it. Hence, we model this facility as Service (A1). The resident-parker is interested in parking her car. Therefore, we model the car as the PE (A2). The car is identified in the physical world by a license plate number. The latter is modelled as a Device of type Tag (A3). In the digital world, the car of a resident-parker is identified with an entry in a white list. We model an entry in a white list as a VE of type Passive Digital Artefact (A4). Entries of a white list are stored in a database that allows accessing the entries in read and write modus. This database is therefore, modelled as a Network Resource (A5). A software application is responsible for mining the

database white list, for instance for verifying whether a specific car is allowed to park in a given city zone, or if it is in unauthorised. This application software is modelled as an On-network Resource (A6). After the resident-parker successfully registers to the Resident PBL Service, her information needs then to be inserted in the white list database of parkers. This results into one Service invoking the other one as depicted in Fig. 11.3.

Time-Parker

Having the time-parker as an additional user of the system adds only one new part to the already described entities in the IoT Domain Model. A time-parker needs to subscribe to the time-parker PBL system. Here, we also model the functionality provided by the PBL system as a Service (A1). The remaining answers steps, viz. A2 to A6, are exactly the same as the ones for the resident-parker.

Enforcer

The enforcer, i.e. the traffic warden, invokes the application on the handheld (A1). The enforcer is interested in a parked car, which we have already modelled as a PE (A2). The car is identified by its license plate number, which we have already modelled as a Device Tag (A3). The handheld has a sensor that reads the license plate number to identify the car. The type of this sensor depends on the deployment and can be, for instance, an RFID reader or a camera. In any case, we model the handheld as a generic device and the sensor as a Sensor Device (A3) that reads the Device Tag. A car which is allowed to park, is identified in the digital world with an entry in a white list. We have already modelled this entry as a VE of type Passive Digital Artefact (A4). The handheld runs software that computes the sensor readings in order to identify the license plate. For example, in case of a Device camera, this software processes the images taken for a license plate. We model this software as an On-device Resource (A5). This Resource is then directly accessible by the user. Therefore, we do not have a Resource-level Service.

Registry Office

In order to feed the system with the newest information of the registered cars, the registry office invokes software to maintain/query the database. This is done by

invoking the same service as resident parkers, but with different user rights (right to delete entries; right to change payment status, etc. We have already modelled this software as an On-network Service (A1). Answers (A2) to (A6) are also the same as for the resident parker.

11.3 Requirement Process and “Other Views”

11.3.1 *Requirement Process*

As discussed in detail in Sect. 6.4, the requirements process generates view requirements. Major inputs into this activity are

- Business goals
- Physical Entity View
- IoT Context View

All three of them have already been discussed in greater detail above, and we are now progressing to the requirements-engineering step.

11.3.2 *Requirements*

Notice that we do not prescribe any particular requirement-engineering process for how to generate requirements. Rather, the IoT ARM offers a set of aids that ease the translation of requirements into architecture features. For the generation of the requirements a wealth of engineering approaches and aids is described in the pertinent literature. Just one example are the Volere requirements templates (Volere 2013).

An abridged list of requirements is provided in Appendix [requirements for concrete architecture].

Notice that for the sake of brevity, the list in Appendix [:requirements for concrete architecture] only contains an illustrative list of requirements that

(continued)

(continued)

shed light on the IoT ARM supported architecting process. In praxis, unabridged requirement lists can readily contain several hundred requirements. Most of the view requirements are related to the fact that this architecture is an upgrade to an existing system (see Sect. 11.1).

In this section we are not simply repeating the requirements in the Appendix, rather we discuss where and how they enter the architecting process.

As explained in Sect. 6.4, we organise requirements along three disjunct topics:

- View requirements
- Design constraints
- Qualitative requirements

The type of each requirement is listed in the second column to the left in Appendix [requirements for concrete architecture]. Let us have a look at each of the requirement types.

Notice that one does not create requirements ex nihilo, rather they are based on business principles (as indicated in the rationales of the requirements in Appendix [requirements for concrete architecture]). Also, the IoT-A Unified Requirements (see Appendix [requirements]) can be consulted for generating requirements for a concrete architecture (see Sect. 6.7).

11.3.2.1 View Requirements

Examples for view requirements are BPL #5 and #14, viz.

PBL #	Requirement type	Description	Rationale	View	Perspective	Functionality group	Functional component
5	View	Communication with the pay-and-display machines shall adhere to the OCT2 standard.	This is the standard used in the current systems. It readily accommodates the new information types to be exchanged with the Control Centre (car ID, permit details). Also, it is an international standard, so it does not conflict with PBL #1. (business principle)	Functional	None	Communication	End to end communication
14	View	Parking white list is provided by the system.	In order to facilitate all the new envisaged usages (PBL for time parker and resident parker, as well as RBL for the parking enforcement) the Control Centre needs to maintain and provide access to a parking white list.	Information	None	Virtual entity	Virtual entity repository

As already stated above, the IoT ARM does not offer any specific support in deriving requirements (besides the inspiration provided by the Unified Requirements; see Sect. 6.7), rather the IoT ARM provides support in mapping the requirements onto the IoT-ARM concepts. This is exemplified in Section View requirements by the yellow-coloured columns. These columns are populated during the initial mapping of the requirements onto concepts used in the IoT ARM. The core concepts used in the IoT ARM are views and perspectives, and these are shown to the far left.

What view these requirements map onto is indicated in the fifth column from the left, viz. the view column. Here we have an example for a functional-view and an information-view requirement. Both of them can – already at this stage- be mapped onto the functional decomposition that was introduced in Sect. 8.2.2. PBL #5 can be mapped onto the End to End Communication FC in the Communication FG, while PBL #14 can be mapped onto the Virtual Entity FG. Mapping requirements at this early stage speeds up the population of the various architecture views with concrete goals.

Notice that PBL #14 is not mapped onto any of the FCs listed in Sect. 8.2.2, rather onto a new FC, i.e. a Virtual Entity repository. This reflects a design choice made, viz. to not include the Parking White List in the VE Resolution FC, rather in its own FC. One of the main reasons behind this design decision is the evolvability of the system. By keeping the white list apart from the Virtual Entity Resolution FC, it is easier to extend and change the system during future version iterations. This mapping is thus actually attributable to several of the qualitative requirements, viz. PBL #4, #11, #13, who all address the evolvability of the PBL system. See more on this in the below Section on qualitative requirements.

11.3.2.2 Design Constraints

Design constraints define constraints in the design of an architecture. An example for this is PBL #3, viz.

As one can see, this requirement is indeed a constraint in that it tells the architecture not to include payment transactions in the architecture, something that is tacitly covered in the IoT Context View (see above), but in order to avoid slips during the architecting process, it is often very helpful not only to state what is within the system scope but also what is outside of the system scope. An example for a design constraint at the reference-architecture level is UNI.071, viz. “A system built using the ARM shall provide standardised and semantic communication between services”. Here, it is emphasised to standardise interfaces. In other words, non-standardised interfaces do not lie within the scope of the architecture.

PBL #	Requirement type	Description	Rationale	View	Perspective	Functionality group	Functional component
3	Design constraint	Payment transactions are out of scope of the system extension	The current parking-management system relies on the payment system by a third party and this modus operandi is not going to be changed in the new, extended version of the system. (business principle)	None specific	None	None specific	None specific

11.3.2.3 Qualitative Requirements

As discussed above and in Sects. 8.3 and 6.9, qualitative have impacts on more than one view. What is not mentioned in these Sections though, is that qualitative requirements can inform the same architectural design decision. In order to elucidate this point let us look at the three qualitative requirements that inform the decision to store the Parking White List in a Virtual Entity Repository instead of Virtual Entity Resolution FC. These requirements are listed below.

Notice that although all of these requirements are mapped onto the Evolution and Interoperability Perspective, none of them is openly mapped onto the Virtual Resolution Functional Component. This is because perspectives by default do not map on one view nor one FC. So how does one map such qualitative requirements? As discussed in Sect. 6.5, the IoT ARM follows the framework of Rozanski and Woods in that it advocates the choice of tactics in order to successfully map qualitative requirements onto architecture descriptions. Furthermore, as discussed in Sect. 6.9, the IoT ARM also provides guidance in terms of the desing-choice process, viz. what design choices are at hand after a certain tactics has been chosen. One of the design choices spelt out (see Sect. 6.9) is to build the architecture out of models and to couple the blocks loosely. In the context of the PBL architecture this design choice was translated into the decision not to store the Parking White List in the Virtual Entity Resolution FC, but rather to create a new FC, viz. the Virtual Entity Repository. By so doing one decouples, for instance, the evolution of the Virtual Entity Resolution FC from the Parking White List during future PBL version cycles, as long as the interfaces between both are kept up to date. In other words, instead of creating strong ties between resolution and the white list, the coupling is rather loose, and the respective FCs can thus evolve independently of each other.

As discussed above, most of the requirements in Appendix [requirements for concrete architecture] are qualitative in nature. This is mostly due to the fact that the business principles from which these requirements stem are behavioural requirements toward the entirety of the system. An example of this is requirement PBL #1, which stipulates that the system shall be deployable in many countries. Such a requirement has repercussions for many views, for instance information view and deployment view, and it is thus of a qualitative nature.

The Table in Appendix [requirements for concrete architecture] features requirements that are mapped onto perspectives that are not part of the IoT Reference Architecture (see Sect. 8.3). Examples for such requirements are PBL #1 (internationalisation and usability perspective) and PBL #2 (regulation perspective). These requirements are not covered in the IoT Reference Architecture (and thus in the design-choice process) because they are not

(continued)

Requirement		Description	Rationale	View	Perspective	Functionality group	Functional component
PBL #	type						
4	Qualitative	The system shall be readily extensible to encompass off-street parking	In a future version of the system, off-street parking might also be offered or the system shall be able to seamlessly cooperate with third-party systems for off-street-parking management. (business principle)	None specific	Evolution and Inter-operability	None specific	None specific
11	Qualitative	The user base for the system shall not be limited to resident parkers and time parkers in future versions of the system.	In the future new business models are envisaged for the resident-parking service. An example is the inclusion of employees of participating companies in the resident-parking pool. By so doing, the customer base for the Control Centre and thus the total revenue that can be generated with the Control Centre can be extended. This will increase the attractiveness of parking services offered to, e.g., municipalities. (business principle)	None specific	Evolution and inter-operability	None specific	None specific

(continued)

PBL #	Requirement type	Description	Rationale	View	Perspective	Functionality group	Functional component
13	Qualitative	Existing system software shall be reused to a maximum.	Decrease the cost and increase the ROI of the enhanced Control Centre by maximum reuse of existing hardware within the system but also outside (compatibility with existing parking-enforcement systems, etc.). (business principle)	None specific	Evolution and Inter-operability	None specific	None specific

(continued)

important for IoT systems. Rather, we were unable to find IoT-specific aspects and these and other perspectives. Notice that this does not mean that one cannot formulate a design-choice process for these perspectives. Rather, the architect is asked to rely on tactics provided in the literature and to formulate here own design choices. More insight on these and other perspectives and thereto related tactics can be found elsewhere in the literature (Rozanski and Woods 2011).

11.3.2.4 “Other Views”**Information View**

The IoT IM details the structure of the information that constitutes a VE and the Service Description of a Service that acts on the VE (see Sect. 7.4.1). In this section we will describe the modelling of these two elements for the PBL use case. Notice that the information view does not cover data formats. These lie within the purview of the deployment view.

Modelling the VE

Following the IoT Information Model (see Sect. 7.4) a VE can have one or more attributes, each having an attribute Name and an attribute Type. In the PBL use case, a VE is an entry in the Parking-White-List database identifying a car that is allowed to use the PBL parking facility. Since we have considered two types of parking cars (the car of a resident-parker and the car of a time-parker), the VE for one car type is slightly different than the other one. The first two attributes (License plate numbers and Parking zone) depicted in Fig. 11.4 are common attributes for both types of VEs, while the third attribute (Parker type) is part of the VE resident-parker. Notice though that we only chose the license plate number as the VE and did not include in the parking zone. This design decision is based on several previous decisions. First, one of the main business principles behind the PBL system is its future extensibility. As discussed in the requirements Section above, there are many requirements that stipulate the evolvability of the PBL system. This, among others, boiled down into loose coupling rather independent FCs. One example for the latter is the choice to introduce a Virtual Entity Repository for the White List in the system architecture. In the information model we drive this modularisation one step further in that we chose a single piece of information, the license-plate number, as the VE. All the other entries in the Parking List are then associated to the VE. By so

doing one can, for instance, include more attributes in future version of the PBL system.

In the following we define each of the three attributes and discuss their decomposition:

- License plate number
 - attributeName: License plate number;
 - attributeType: Car information;
 - Description: It is a common attribute for both VEs. In essence, it is a numerical and alphabetical registration identifier that officially and uniquely identifies the car within an issuing region such as the entire country or entire state;
 - Values: A resident-parker or a time-parker may own and may have parked more than one car on the street. Therefore, it is necessary that this attribute has one or more Values, each one containing the License plate number of a registered car. The example in Fig. 11.4 states the registration of two license plate numbers: “M CJ 1234” and “M JW 5678”.
- Parking zone
 - attributeName: Parking zone;
 - attributeType: Parking information;
 - Description: It is a common attribute for both VEs. This attribute identifies the parking zone, where this car is allowed to park;
 - Value: corresponds to the name or the identifier of a parking zone. The example in Fig. 11.4 depicts the registration Zone A.
- Parker type
 - attributeName: Parker type;
 - attributeType: Parking information;
 - Description: It is an exclusive attribute in the VE of a resident-parker. This attribute identifies the time during which, this car is allowed to park. Currently we differentiate between a full time parker (24/7) and a night parker (12/7);
 - Value: it is either 24/7 or 12/7. The example in Fig. 11.4 shows a registered night parker.

For the sake of clarity, we note for the reader that other attributes can be added for each VE as well as other Values and MetaData. These highly depend on the deployment of the PBL facility, which definitely changes e.g., from one city to another.

Fig. 11.4 IM of the VE for resident-parker and time-parkers (*Orange edge*: unique to resident-parking)

11.3.2.5 Functional View

Technical Scenarios

Besides the rather static descriptions provided by the Physical Entity View, the IoT Context View, and the business goals (see previous Sections), we have found that the semi-dynamic view of UML use-case diagrams is very helpful in the identifying salient FCs in the functional decomposition and also the interactions and interfaces of said FCs. Below we provide use-case diagrams for all major technical use-case scenarios of the PBL system.

Notice that the system-boundary boxes in the use-case diagrams are not synonymous to the boundary of the PBL system. Rather, they allude to entities in the context view (see Fig. 11.2). All thick-lined boundary boxes are part of the PBL system (see Fig. 11.3).

Purchase (and Change) of Parking Permit

See Fig. 11.5

This diagram summarises how the time-parker interacts with the Control Centre. It has implications for manipulations and thus the interface of the Virtual Entity Repository, but also for the VE Resolution, because in order to extend a permit it first has to be located in the system (Fig. 11.6).

(continued)

Fig. 11.5 Technical use case – purchase of parking permit by time-parker

Fig. 11.6 Technical scenario – subscribe/unsubscribe/change by resident-parker

(continued)

This technical use case summarises how the resident parker interacts with the Control Centre. Notice that the actions on the primary-service level are not part of the PBL system. It has implications for manipulations and thus the interface of the Virtual Entity Repository, but also for the VE Resolution, because in order to extend a permit it first has to be located in the system.

On-Street Parking

This technical use case summarises the actions triggered when time-parkers and resident parkers actually park their car. Since the time-parking scenario is of an ad-hoc nature (all pertinent actions conducted shortly prior to or during parking), while the resident-parking scenario is of a recurring nature (payment of fees, etc. well in advance to individual parking events), the former incorporates many more use cases than the latter. This technical use case has implications for the interface of the VE FC and the Security FC and the interface the PBL exposes toward the PDM/webserver (Figs. 11.7 and 11.8). This use case has no implications for the architecture.

Parking Enforcement

See (Fig. 11.9)

This use-case diagram summarises the parking-enforcement scenario. Notice that the “get licence plate” use case includes the parked car as an empty system. This technical use case has implications for the interface of the VE FC and the Security FC and the interface the PBL exposes toward the PDM/webserver.

Fig. 11.7 Technical scenario – on-street parking by time-parker

Fig. 11.8 Technical scenario – on-street parking by resident-parker

Fig. 11.9 Technical scenario – parking enforcement

Modelling the Service Description

Following the mapping of the IoT DM to Service Description explained in (Martín 2012; Sect. 4.6.3), we model the VE-level IoT software application for mining the Parking-White-List database. Notice that multiple other software applications may act on attributes of VEs and can be modelled as well. Examples of these software applications are updates of attributes; running statistical inference on VEs; applying mathematical operations on VEs; and representing attribute values on graphs and charts. Here we focus on the modelling of the mining software.

Fig. 11.10 Service description for the PBL system

Figure 11.10 depicts the Service Description of the mining software. In the following, we will highlight the service specifications:

- **hasServiceArea:** This service runs in the Control Centre that is generally responsible of managing PDMs in a single city. Consequently, all parking zones in a city are affected by this service;
- **hasInput:** In order for the mining service to verify if a car is allowed to park, the enforcer needs to provide the service with three input data: The current time, the geographical location or the zone of a parked car, and the license plate number of the parked car;
- **hasOutput:** Having the three aforementioned input data of the parked car, the mining service verifies all the VEs in the Parking-White-List database. After the verification, two results are possible:
 - The given license plate number is matched in one of the VEs: In this case, the service compares the given geographical parking place and parking time with the corresponding attributes of this VE. If the car is not allowed to park at this place and/or at this time, the service decides that it is a violator. Otherwise this car is allowed to park;
 - The given license plate number is not found in the database: In this case, the service decides that the parked car is a violator.

Exposes: As previously explained in the domain modelling of the PBL (see ARM document Sect. [11.2.3](#)), this service exposes the Parking-White-List database as Network Resource.

Open Access This chapter is distributed under the terms of the Creative Commons Attribution Non-commercial License, which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.