Requirements Analysis for Smart Citizen Assistant:

Use Cases and Scenarios

Deliverable D4a.1

Version 0.2

Responsible Partner:	Siemens AG Österreich
Authors:	Josef Aigner Stefan Bischof Deepak Dhungana
Work package partners	Siemens, ETA, Wien 3420, Energiecomfort, Wiener Netze, MA18, MA20, MA21, ÖIR

Executive Summary

This document describes a set of challenges and opportunities associated with a new application programming interface (API) for a smart citizen assistant (SCA)—a platform for diverse smart city applications based on data available in a city. We present a set of API features and implementation technology selection criteria, identified by considering two use cases and API usage scenarios.

An API is a software-to-software interface, not a user interface. With APIs, software applications talk to each other without any user knowledge or intervention. The goal of defining the SCA is to provide the software developers and designers a set of easy to use abstractions when developing new applications. Here, software developers and designers are the "end-users" or the "customers" of the API. The primary goal of the API is to offer data to the developers in a controlled, cost-effective and user friendly manner in order to maximize user experience and operational efficiency. It enables other smart city stakeholders to visualize data without increasing the complexity of accessing and integrating data that comes from different sources. In this sense, the API indirectly helps in raising awareness of environmental topics in a smart city, because different applications targeted to different customer segments (i.e., demographic aspects, social status or educational background) can exploit the data through the API. Although the end user applications are not in focus of this project, example usage of the API will be demonstrated using a sample APP, which illustrates how data from multiple sources can be accessed in a seamlessly homogenous manner, thereby raising the level of information content and enriching the semantic of the data.

The two use cases chosen for requirements analysis are very diverse and address completely different perspectives on the data. This ensures the flexibility of the API and makes its design extensible for other future applications, which will emerge after the API is available. By considering the perspective of infrastructure provider and the end-users, we serve two primary groups of application users.

The biggest challenge is the change in philosophy of how data is seen by software developers – there is a paradigm shift from "silos" of data from one particular domain to a flexible API for a modern, web- or mobile applications, where data from multiple sources in a smart city are integrated and presented to the end-user in a unified manner. Major improvements in terms of better end-user experience can be achieved through the use of state-of-art technologies and a change in mind-set of developers. Another major challenge lies in the fact that the products and services for different kinds of end-users are quite different in terms of size of data set, number and frequency of API calls and privacy concerns. The acceptance of the new API is likely to be very high as it opens up new horizons for application developers, giving them access to integrated data from different "data-silos" in a city.

This document only covers the technical aspects of designing the API and some background information on the project landscape it fits in. The focus lies on the technical feasibility of the approach and not on legal, social or financial issues around data management and user management. These aspects are out of scope of this project.

Contents

List of Figures

Abbreviations

API	Application Program Interface
ASCR	Aspern Smart City Research
BEMS	Building Energy Management System
BI	Business Intelligence
CSV	Comma Separated Values
DB	Database
DC	Data Concentrator
EJB	Enterprise Java Bean
ETL	Extract, Transform and Load
JMS	Java Message Service
JSON	JavaScript Object Notation
KB	Knowledge Base
MDM	Meter Data Management
MDMS	Meter Data Management System
MQ	Message Queue
PLC	Programmable logic controller
ReST	Representational State Transfer
SCA	Smart Citizen Assistant
SG	Smart Grid
SUL	Smart Urban Lab
UDA	Unified Data Architecture
UI	User Interface
XML	Extensible Markup Language

1 Introduction

Transform+ is a research project funded by „Klima- und Energiefonds“, as part of the Austrian research funding agency FFG. The goal of the project is to prepare and operationalize the contributions of the EU project „TRANSFORM“ in the city of Vienna. One of the outputs of the project will be implementation plans for pilot projects in "Liesing-Groß Erlaa" and "aspern Seestadt" (Smart Urban Labs).

As part of this project, a "Smart Citizen Assistant" is to be designed for Aspern Urban Lakeside, which is a platform for visualizing smart city data to the residents. The aim is to provide a novel interface through which relevant data can be accessed individually and in a timely manner.

1.1 Purpose of the Document

This document illustrates the scenarios in which the smart citizen assistant will be used and analyses the requirements for application programming interface (API) design based on the use cases. The use cases presented in this document **are merely examples** of how the API could be used. The API is **not restricted** to only these scenarios; furthermore, not all aspects of the API are exemplified by the use cases.

As part of Transform+, the requirements identified in this document will be considered during design and implementation of the API. The practical feasibility will be demonstrated (towards the end of the project) by feeding data to an existing APP through the API.

1.2 Context

In order to understand the use cases around the smart citizen assistant (SCA)[1], it is important to understand the project landscape and the context in which the SCA is being developed. It is part of the initiative “aspern Vienna’s Urban Lakeside”, which is a project initiated by the city of Vienna and it is one of Europe’s largest urban development projects.

[1] The smart citizen assistant (SCA) will be used as a synonym for the data access layer in the data warehouse, which means it will cover both the end-user perspective and the infrastructure perspective.

The goal is to design and develop a novel interface that makes it possible to access relevant data (in this case electricity data) individually and in a timely manner for personalized, clear and understandable use. The extensible design of the interface **allows the future integration** of new data sources and the extensions of the core functionalities as well flexible integration of new and existing applications.

Currently, it is of course difficult to define which data will be relevant for a certain application in the future – therefore, the API will be generic enough to provide available data in a flexible manner. The API can be configured and adapted to be application specific, when it is clear which data is needed.

1.3 Smart Urban Lab (SUL) Aspern

In conjunction with partners from the city of Vienna, Siemens is launching a large smart city project in Vienna, Austria. A living laboratory will be created in the next five years in the water-side district of Aspern, one of the largest urban development projects in Europe. Here power supply, building systems, intelligent power grids, and information and communication technologies will interact optimally. The result will be the most efficient resource management possible, with maximum comfort for residents and users. As the infrastructure in Aspern will be built over a long period of time, the design of the API and the applications based on these APIs will not be able to exploit the full infrastructure from the very beginning. The API will therefore have to grow with the available infrastructure components and available data sources.

This project represents an opportunity to develop a long-term integrated concept for an energy-optimized city district using appropriate technologies, products, and solutions in a real-world infrastructure. The goal is to make the whole system "smarter." One step involves connecting buildings that have different functions, i.e. offices and apartments, to the low-voltage distribution network. In the future building control systems will manage the energy exchange between buildings and optimize energy consumption locally. This offers building operators the possibility to participate actively on the energy markets.

Information and communication technologies play an important role in this process, as does data evaluation. New IT solutions detect faults in the system, recognize inefficient consumption patterns, and identify potential opportunities for savings. Decentralized power generation from renewable energy sources will supply Aspern's electrical needs. Modern storage technologies will play an important role here.

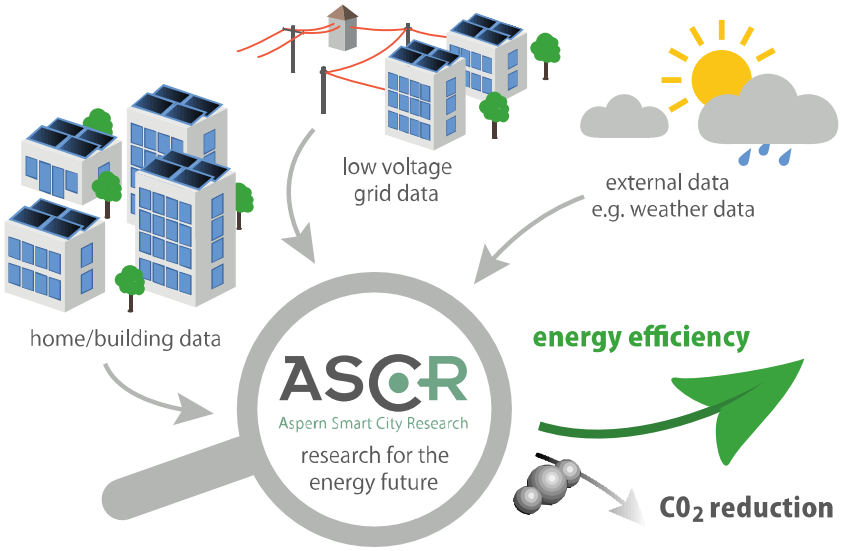

Figure 1 Overview of the research activities and research goals of ASCR

Innovative energy distribution systems can improve reliability and reduce costs of energy supply, but the real value of a smart grid can be exploited only in very close interaction with smart buildings. Therefore, the R&D activities of the proposed project focus on interaction of smart buildings in smart grids, with the goal of enabling participation of buildings in energy markets. With the SUL in Aspern, it will be possible to experiment with different models of operation to gain insights into the optimal configuration – i.e., required components in the buildings/grid, and ICT methods and tools for data analytics and monitoring. This is therefore the focus of the research program.

In the context of Transform+, this means, that the citizens of Aspern will benefit from the research activities as they will have novel means of influencing their energy consumption. The motivation of the users should be increased with innovative ways of presenting the data and exploiting this to reduce financial burdens of the end-users. This aspects are important to consider, but do not directly correlate with the design of the API (rather with the design of the APP). Therefore, research on how the end-users can be motivated is out of scope of this project.

1.4 Overview of the Document

This document is structured as follows: In Section 2, some background information is presented to set the context for SCA, in the project landscape in and around seestadt Aspern. In Section 3 and Section 4, two use cases are presented in which the usage of the API is exemplified on two different kinds of applications. In Section 5, the use cases are analyzed and requirements are identified. We conclude the document with some conclusions in Section 6.

2 Smart Citizen Assistant (SCA)

The Smart Citizen Assistant prototype provides a standard interface for applications for mobile devices such as smart phones or tablets. We use SCA as a synonym for the data access layer in the data warehouse, which means it will cover both the end-user perspective and the infrastructure perspective. Users will be able to compile an individual list of applications of diverse domains (e.g., energy, mobility, communities) and thus get their personal portal to relevant data of "their" smart city. Existing successful applications can be integrated in the future. With increased usage the Smart Citizen Assistant can be adjusted to the demands and needs of the user. Effective data structures and interface specifications are the main preconditions for developing apps for the Smart City Assistant. This project defines the *interfaces* usable for future applications.

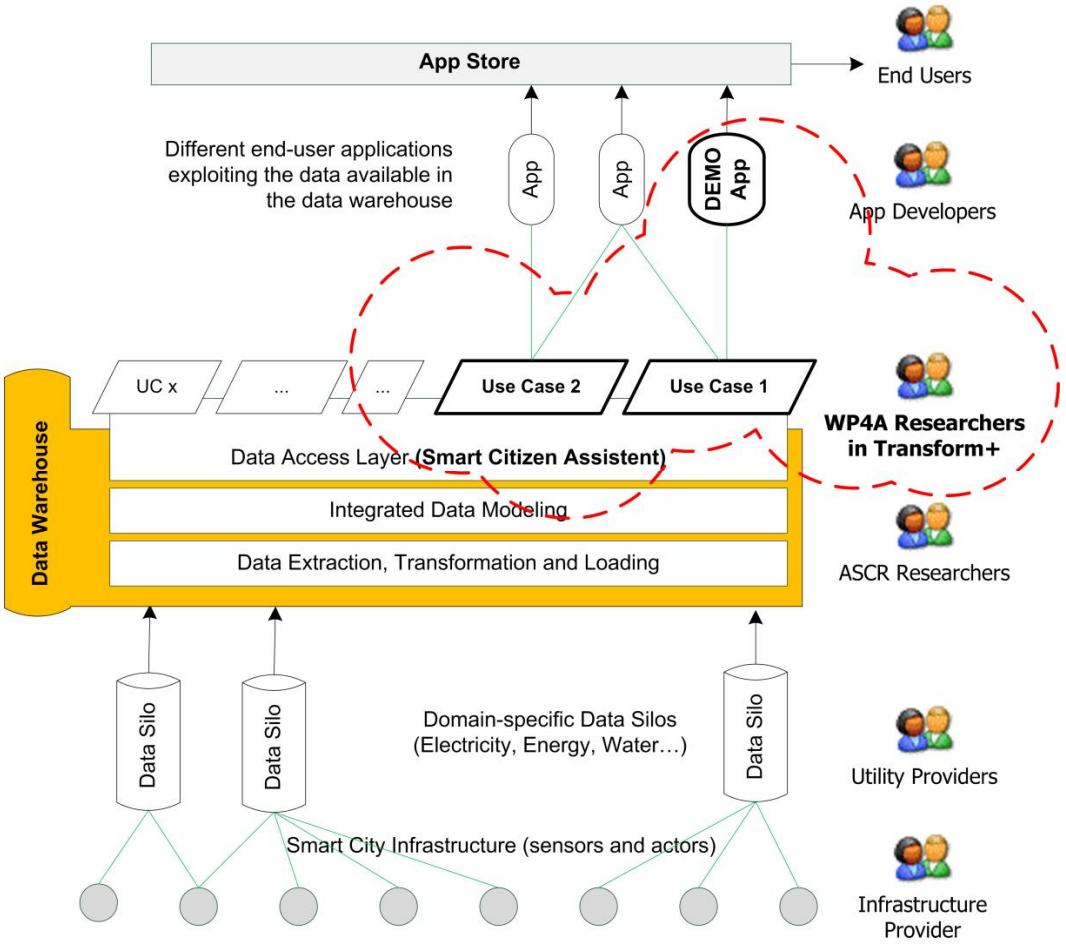

Figure 2 Position of the API and its relevance for the Transform+ Project depicting the involved stakeholders and their roles. The dotted line shows the activities in Transform+

In this document, we describe two different use cases for the usage of the data from the data center. This represents only a small part of the Data access layer of the overall data warehouse. Further innovative data access facets will be defined in the ASCR project, where more use cases are handled and more apps will be developed for other stakeholders. In Transform+, we test the API and other technical components through a DEMO app, which visualizes user specific information about energy usage. The user specific information is presented in an intuitive way and gives users an overview of their energy usage data (electricity). Additionally environmental impacts of a user's actions are presented together with tips and information on how to decrease electricity usage.

In order to demonstrate the added value of integrating multiple sources of data, the DEMO App will have new functionality that goes beyond the features already available in the smart metering APP of Wiener Netze. The added value could be that the APP combines smart meter data and some external data source (e.g., weather data) to gain new insights.

New application (scenarios) such as public transport, car sharing could be added successively as new elements to the SCA. This is however not part of this project—nevertheless, the technical infrastructure will be able to easily handle such extensions.

2.1 API Design

An application-programming interface (API) is a set of programming instructions and standards for accessing a Web-based software application or Web tool. A software company releases its API to the public so that other software developers can design products that are powered by its service. For example, Amazon.com released its API so that Web site developers could more easily access Amazon's product information. Using the Amazon API, a third party Web site can post direct links to Amazon products with updated prices and an option to "buy now."

The data that is available in the context of a smart city gives users access to applications and content on huge variety of connected devices in homes, offices, cars, pockets etc. The glue that allows all of this to happen, i.e., that connects the parties who provide these data services to the devices that end users can use, is the **API**. That's why APIs have a huge responsibility for many developers (and companies), and so it is natural that API design touches on a range of topics including resource modeling, payload format, how to version the system, and security. While these are likely important areas to explore when designing virtually any API, the reality is that a much larger decision needs to be made first. That decision is based on a fundamental question: who are the primary audiences for this API and how can we optimize for those audiences?

A proper analysis of the use cases and the usage scenarios are important for API design, as APIs can be among a company's greatest assets mainly for three reasons, i.e.,

- Customers (developers) invest heavily: buying, writing, learning the API
- Cost to stop using an API can be prohibitive
- Successful public APIs can involve and engage end-users

A good API must be easy to learn, easy to use (hard to misuse) and sufficiently powerful to satisfy requirements. As the requirements constantly change over time, it should also be easy to extend. Another guiding principle for good API design is that it should be appropriate to audience, meaning it should consider the goals of the stakeholders that it is targeted for. Different experience and background of the users leads to different perceptions on what is intuitive and what isn't. An API is intuitive if a semi-experienced user gets away without reading the documentation, and if a programmer who doesn't know the API can understand code written using it.

Within the scope of Transform+, the API for SCA will be designed by considering the perspective of two different stakeholder groups on the same data. This is particularly important, as the queries, formats, granularities and abstraction level of the data heavily depends on its purpose.

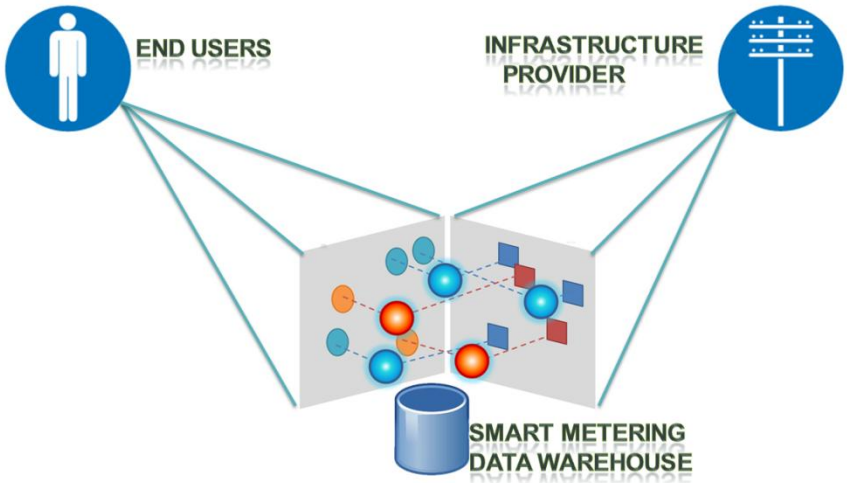

Figure 3 The API will be designed to access data for different applications: based on the usage scenarios from the perspective of end users and the infrastructure providers.

No matter, which kind of data is served to the APP, the features of an API has huge impact on performances. As no users will particularly like an unresponsive APP, the API should serve data in a comprehensive manner. When the API is too lazy or incomplete, the burden is put on the mobile app. An API is also a contract between the APP developer and the data provider that dictates, what can or cannot be done directly. The API must be consistent—meaning developers should know what to expect, when calling an API (also called the principle of least astonishment).

2.1.1 Developing Apps for End-users

End-users are typically interested in monitoring their private energy consumption and visualization of data over time. Developing apps for such a context must consider data of the individual users, their privacy settings and the history over longer periods of time. See Section 3 for details on this perspective.

2.1.2 Developing Apps for Infrastructure providers

Infrastructure providers are interested in utilization of standard 3rd party meters for grid monitoring to generate load flows for grid state estimation. This means, developers of such apps must consider consolidated, aggregated data and analyze these in the light of how it can be used to maximize grid efficiency and optimize LV grid operation. See Section 4 for details on this perspective.

2.1.3 Other potential perspectives on smart city data

In the long run, it is possible, that the data needs to be accessed by third parties (e.g., statistics department), where the key requirements are on the aggregation and anonymization of data to make these publicly available. This perspective represents however, for the sake of API design a sub set of features required by the end-users and the infrastructure providers.

2.2 Technology Considerations

The decision about which technology platform will be deployed for implementing the data center in aspern Seestadt has already been taken. The data warehouse will be based on Teradata, which offers analytic data platforms, applications and related services. The SCA of Transform+ will then use the generated information as described in the use-cases in Sections 3 and 4 below.

Teradata consolidates data from different sources and analyzes data to generate new insights. Teradata's Unified Data Architecture™ (UDA) is a secure and in principle vendor agnostic framework for smart data management, processing, and analytics.

Figure 4 [1] shows the high level logical view of the UDA. Data flows from left to insights to action. We will now have a more detailed look at all the different logical or functional blocks of

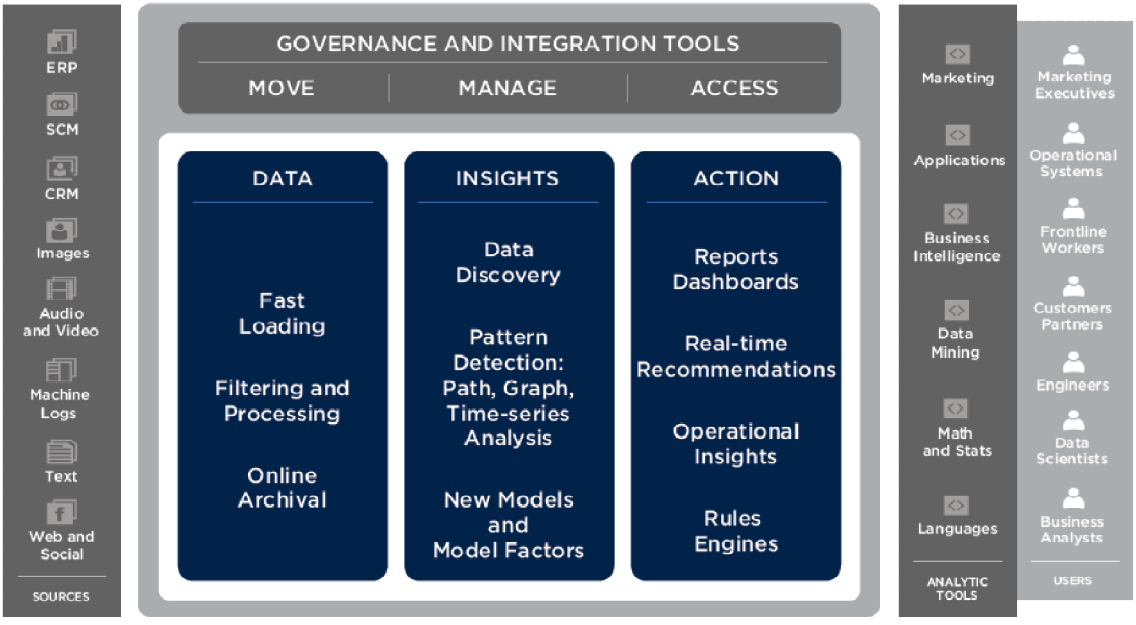

Figure 4 Data warehouse from Teradata enables integration and linkage of data from multiple sources, so that app developers can use the new insights to create future applications. [1]

the UDA.

Sources. Data is collected in different form and from different kinds of systems and data sources on the left. Additionally to classic data warehouse approaches possible input date sources include not only highly structured data sources but also semi-structured data and unstructured data as well. Structured data is usually available from the web and legacy systems such as ERP, CRM, or other corporate management systems. Semi-structured data from different kinds of sources such as public internet data, software services of a service oriented architecture (SOA). Similarly to other common Big Data implementations unstructured data such as images, audio, video, and text data can also be imported and managed with UDA implementations. The DATA block specifically includes Extract-Transform-Load (ETL) approaches and technologies known from data warehouses.

Data. In the first block the UDA provides data management services. At this stage data retrieval, cleaning, filtering, and preliminary processing is performed. Data archival is also provided in this step allowing for referring to historic data as quickly as possible.

Insights. The insights block provides different kinds of tools to generate new insights off the collected data. Through exploratory data analytics patterns in the data can be unveiled. Different kinds of pattern detection tools can be applied.

Action. The generated insights are then communicated to different kinds of users by using several tools. Reports and dashboards are well known tools for communicated complicated related data. Push-style communication such as alarms or messages is another possible means to communicate to affected systems and users.

Analytic Tools. The generated insights and the communication via the action toolkit can afterwards be used by external analytic tools. These tools can be automatically running applications or algorithms. Also interactive tools directly controlled by a specific user group can be used at this stage.

Governance and Integration Tools. Accompanying the whole data-insight-action functions is a toolkit for controlling for governance and integration functions. These functions are not only needed for managing and accessing data and insights but also for enforcing policies.

For the planned smart citizen assistant, this means that the data sources that may be relevant are not necessarily integrated from the start, but can be added in the future. The architecture allows processing all different kinds of data without constraining possible data formats or data representations. UDA provides a component to generate new insight and can thus give users more information by summarizing and explaining the relations to data from very different sources as remarked later in Section 5.

The governance and integration tools can also play an important role in the future to enforce data privacy laws and regulations.

3 Use Case 1: Developing Apps for End-users

Electricity usage information, be it current or historic, are interesting for consumers for different reasons. End users often want to reduce costs or help the environment by minimizing electricity usage. Currently consumers have to wait for the bill of the electricity operator to comprehend their usage. In the end, users get a single number (the electricity usage) describing their usage for a whole year or a quarter. This kind of presentation is hard to comprehend and a user might have no idea why his/her usage increased/decreased so much in the last billing cycle.

An appealing app for smart phones and tablets can help end-users achieve the goal of minimizing electricity usage more easily. Smart meters measure electricity usage at least daily and transfer the measurements to a central server (see also Meter Data Management System in Section 5.1.1). The new meters have to main advantages for the end user:

- More fine granular data: at least one measurement per day, and
- Check usage data much earlier: usage measurements available after about 24 hours.

By being able to access these finer granular measurement data in timelier manner users can understand their usage more easily. By also allowing the smart meter to collect several measurements over the day (e.g., every 15 minutes) users are even able to analyze their own usage patterns over the day. These usage patterns can give users an indication where to look for energy hogs.

3.1 Scope and Objectives

This use case aims specifically at end users and their demands and needs as outlined above.

End user applications, be it web applications or phone apps, must be easily understandable, even for users who use them very seldom. To enhance usability a simple presentation should be favored to technically exhaustive and complicated dashboards.

Mobile apps or websites are prime media for this kind of information because they deliver up-to-date information anywhere and are easily accessible by common devices such as smart phones, tablets, or laptops. These kinds of applications are also well supported by mature libraries and toolkits for visualizing and presenting data.

It is the objective of the API to deliver the data in a secure and timely manner. But it is not the objective of the API to aid in visualizing data or favor certain tools for doing so.

3.2 Scenario: Request Historical Data

The user starts the metering app of the SCA. Login information are either stored safely by the SCA or entered each time the app starts. Since the login mechanism will protect the privacy of the users' data the SCA should use an up-to-date and secure method.

The metering app downloads the most recent data and (depending on the options set by the user) the historical data as well. To address privacy concerns, the user can decide if the app should store historical data on the phone for faster data access (and reduced app startup time), or if all the data should be deleted when the app finishes. The user then is presented an overview visualization of the most recent electricity usage measurements. Standard timeline charts designed in an appealing way make sense as default visualization. More sophisticated visualizations giving the users more insight, e.g., providing data analytics functions to comprehensively analyze electricity usage, might be planned for the future.

3.3 Scenario: Compare User's Data with Reference Load Profiles

The app is started as in the scenario above. After showing the electricity usage data, the user can select a reference load profile to compare his usage to. Different reference load profiles will be available, e.g., *two person household*. By comparing the users load profile with a reference load profile, a user can better evaluate his own usage and might be motivated to save energy, e.g., by upgrading appliances.

3.4 Stakeholders

Actor Name	Actor Type	Actor Description
End user	Person	The end user wants to track individual electricity usage as well as compare usage to standard load profiles.
Phone App	System	The phone app displays individual electricity usage as well as comparison load profiles on a smart phone.
Web Application	System	The phone app displays individual electricity usage as well as comparison load profiles in a standard web browser.
Meter Data Service	System	The Meter Data Service implements the API and provides electricity measurements for individual users

Different kinds of extensions of this scenario are possible. The API will provide data access to reference load profiles. Possible extensions can visualize this data, or connect with other data sources or services. The app as well as extensions has the potential to help the consumer save energy in different ways, while the API should not over constrain future developments.

4 Use Case 2: Apps for Infrastructure Providers

The data collected from smart meters is not only interesting for the end-users to understand their consumption and energy saving potentials, but also for the infrastructure provider to monitor the low voltage grid based on metering data. See Figure 5 for an architectural overview of the backend for measuring, collecting, storing, and analyzing usage measurements.

4.1 Scope and Objectives

This use case takes the perspective of the infrastructure provider and describes how the data could be potentially used for grid monitoring. Data related to load profiles and additionally available metrics from commercially available modern smart meters could be used to detect critical states of the grid and plan extensions to the grid as an reaction to avoid similar situations in the future.

4.2 Narrative Description

On a daily basis, the application requests/accesses the load profiles and other metrics (e.g., voltage metrics) from the smart meters. This data is passed on to other applications that can perform simulations and load flow calculations in an automated manner.

Apart from the 15-minute meter reading from the smart meters, other data such as the corresponding voltage per phase of the meters and phase difference of the corresponding feeders in the transformer stations build up the input parameters of the simulation.

The goal of the simulation is to predict precisely and estimate the grid operation parameters at the point to delivery to the end-customers. Whenever any abnormal fluctuation or critical state is detected, alarms can be raised by the low voltage grid control system. Apart from that, the applications assume that the following data is persisted and available from the data warehouse:

- Pre-calculated compensatory values for the effect of loss through wires, which need to be considered by the simulation tools in order to estimate the grid parameters at the point of delivery.
- Corrective measures to detect and interpolate the potential errors in measurement by the end-devices.

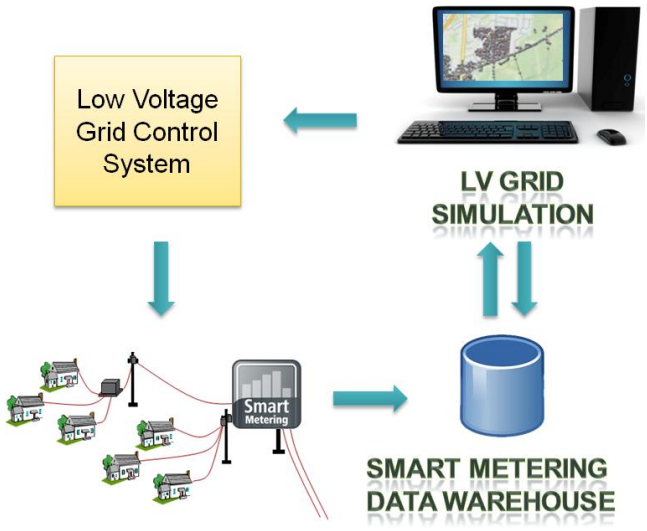

Figure 5 The usage of the API for developing APPS relevant to the needs of the infrastructure providers.

4.3 Stakeholders

Different systems and devices are involved in the collection and persistence of the data, on the basis of which new applications for infrastructure providers can be developed.

Actor Name	Actor Type	Actor Description
Smart Meter	Device	The main task of the smart meter is to measure the energy consumption of end-users according to the legal regulations active at that point in time. Additionally, some commercially available smart meters can deliver more information about the grid if needed, for example, voltage measurements per phase, which can be used to estimate the grid operation parameters for simulations.
Data Concentrator	Device	The data concentrator is responsible for aggregating the data from Smart Meters, under the circumstance that PLC or meshed radio is used for the communication.
Meter Data Management	System	MDM is a system for validation, persistence and analysis of large data sets coming from smart meters.
LV Grid Simulation	System	The Simulation application is responsible for estimating the state of the grid based on the data collected from the Smart Metering Systems.
Device Manager	System	Responsible for automated calibration, configuration and administration of the sensors.

Low Voltage Grid Control System	System	"Operating System" for the grid to administer the different components (Low voltage SCADA, alarm handling, workforce management etc). This system is able to determine and maintain the topology information about the LV grid.

4.4 Step by Step Analysis

The data is sent through different applications, aggregated, analyzed, and saved for later use. The following table depicts the typical steps involved in this process.

Step	Event	Actor Description	Information Producer	Information Receiver	Information Exchanged
1	Request Measurement	MDM requests on a daily basis, the measured data of the last 24- hours.	Smart Meter	MDM	Load profiles
2	Send load profile data	MDM sends the load profiles and other metrics (depends on the manufacturer)	MDM	Low Voltage Grid Control System	Load Profile and Grid Parameters
3	Transfer Energy Data	Load profiles and additional metrics are sent to the simulation system	Low Voltage Grid Control System	Simulation System	Energy Data
4	Load Flow Calculations and Alarms	The Simulation application calculates the load flow and alarms if critical states are reached	Simulation System	LV Grid Control System	Alarm
5	Persist Results	Low voltage grid simulation systems sends the data to the data warehouse for persistence	LV Grid Control System	Data Warehouse	Data for long-term persistence

Based on the different steps involved and the kind of data sent through the different APIs, it is now possible to analyze the steps to deduce the requirements for the smart citizen assistant API.

5 API Requirements Analysis

An application programming interface (API) specifies how some software components should interact with each other. API design is very much like user interface and user experience design. The target audience (developers) has different needs and characteristics. Similar to a friendly, usable app user interface (UI), the API should also be designed to be intuitive, forgiving and frictionless. The goal of this section is to allow an organization that designs the API to access data from a smart city data center make sure the needs of developers and applications using the API are being met.

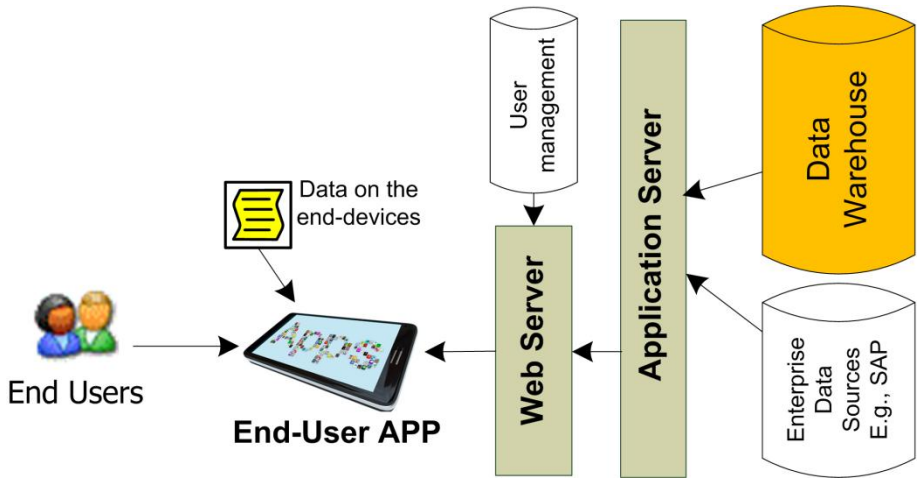

Figure 6 A typical architecture of a data-intensive mobile APP, showing the diverse sources of data playing a role in providing the end-user experience.

We base our discussion on the typical requirements for API analysis on a typical architecture of a data-intensive mobile APP depicted in Figure 6. In the end, the APP needs data from multiple sources, these data sources serve different purposes (e.g., customer databases in SAP, user registration management in a web server), which are external to the data warehouse. This kind of data **must be maintained outside the data warehouse** and is therefore not available through the Smart Citizen Assistant API for several reasons:

- **Data is specific to the APP** and thus cannot be shared via the data warehouse: This includes data such as user registration and rights management in a web server.
- **Data is highly critical** and should not be made public: This includes data on customer contracts, fiscal transactions, bill management etc which is typically available in enterprise data management systems such as SAP.
- Some other **data can only be saved in the end-devices** of the user and must be deleted after usage—mainly due to legal and regulatory issues around the different kinds of data.

As a summary, we conclude that the data sources that need to be considered during the API design are only data sources that can be integrated to the data warehouse. The requirement for the API is thus to provide an efficient data access to the data in the data warehouse.

5.1 Data Sources

During the implementation phase of SCA, we will use demo data, since "real" data is not available. The prototype will be tested and reviewed by selected users. The users' feedback will then be used later for future enhancements of the Smart Citizen Assistant.

In the field, all required data has to be collected from different kinds of sources. The SCA, in this case similar to the Wiener Netze Metering APP, uses the data provided by the ASCR Teradata Data center. The ASCR Teradata Data Center collects, stores, and joins the data from these sources. The data center thus organizes the data from different heterogeneous data sources in a global integrated schema.

5.1.1 Meter Data Management System

A Meter Data Management System (MDMS) collects, stores, and verifies electricity usage data delivered by smart metering systems. These systems are usually operated by electricity network operators. The current MDMS of Wiener Netze powers their own Metering App. The ASCR will use a separate MDMS for the Aspern project containing only measurement data from the aspern LivingLab.

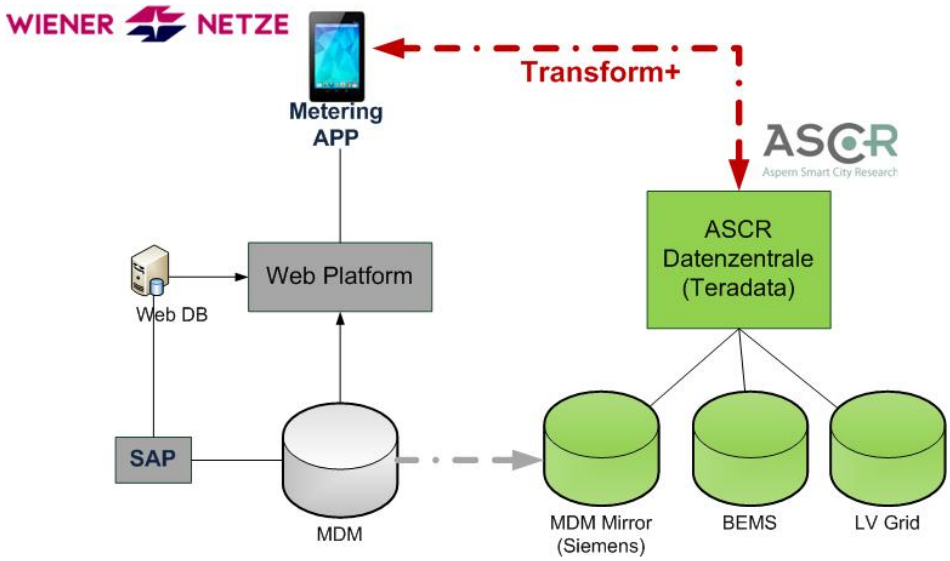

Figure 7 Technological context in which the API will be used: Teradata is used as the database technology in ASCR research projects.

5.1.2 Low Voltage Grid Control System

The low voltage grid control system might also provide data for the ASCR Teradata data center. These systems are mainly concerned with monitoring and regulating the low voltage grid in

urban areas. The control system is mainly used by the electricity network operators internally and, if at all, only interesting to business users.

5.1.3 Building Energy Management System

A Building Energy Management System (BEMS) is an IT system which monitors and regulates several building services such as heating, air conditioning, ventilation, lighting, or blinds. Each building would use its own BEMS. For increasing energy efficiency the BEMS data is interesting for business users as well as private users. Since ASCR data center provides only read access the SCA too, can only provide read access and cannot allow any regulating actions.

5.1.4 Other publicly available data

There are several sources of data for other purposes (e.g., weather data, GIS data, data from the city, world bank data[2] or the UNO data API[3]) which are publicly available through different organizations, who have committed to open data policies. These data sources can also be included in the data warehouse, if needed, to generate some added value for the developers in providing new functionality in the apps to be developed.

The City of Vienna[4] has committed itself to the concept of Open Data - an open and transparent system that makes city data available to citizens for their further use. The present data catalogue comprises over 200 machine readable data records, and is being expanded continuously. In addition to statistics (such as population statistics) and geographic reference data (e.g. on collection points for recyclables, kindergartens, short-term parking zones and hospitals), the first edition of the catalogue already includes selected budget data on the city's annual accounts and financial management. There is also a free online city map of Vienna with more than 120 layers (i.e. levels showing specific geographic information, such as the location of pharmacies, kindergartens or one-way streets.)

5.2 Data Quality

Data Quality is the step that involves taking raw data and cleaning so that we have a base data set that satisfies certain fundamental quality criteria. If we are concerned with semi-structured data, quality may encompass things like making sure we have addresses that are valid and complete, names are sensible, numbers that are expected to be positive are positive, garbage rows are removed, etc. It can also include doing entity resolution and record linkage so that we get a more accurate picture of what is going on.

Whenever data needs to be processed automatically, then data quality becomes an issue. In the usual data integration scenarios, in this case the ASCR data center, this problem is aggravated because data of data providers with very different concepts of data modeling, granulari-

[2] http://data.worldbank.org/
[3] https://www.undata-api.org
[4] https://open.wien.at/site/

ty, and especially quality needs to be combined. This data integration task is one of the main functions of the ASCR data center.

Nevertheless some data quality issues will carry over to the API and therefore also to the SCA. SCA applications thus need to handle the following data quality issues:

- **Missing data** If necessary inform the user of missing data
- **Erroneous data** If possible mark suspicious data values

When designing an API, it is important to foresee methods for app developers to deal with inadequate quality data. For example, the application should not crash simply because the API call returned some unexpected data or the data is missing.

5.3 Data Representation

Data representation is an important factor, when defining an API. The goal is to be flexible in terms of data format and the choice of technology in accessing the data. A growing open API movement, headed by big name companies like Facebook, Google and Twitter, has led to reduced API dependency upon conventional heavyweight service-oriented architecture (SOA) in favor of more lightweight JSON and REST services. Some API management tools are capable of converting existing SOAP, JMS (Java Message Service) or MQ (Message Queue) interfaces into RESTful APIs or JSON content.

The API of the smart citizen assistant should be flexible enough to deal with different formats, ideally, the developer is able to configure the API to suit her needs. However it is important to define, fix and finalize formats for meta-data and binary data formats (if needed). This decision will also be heavily influenced by the choice of the data ware house technology (Teradata).

5.4 Data Access Rights Management

Based on the different perspectives on the data in the data warehouse, it is important to differentiate between different kinds of data (personalized, aggregated, anonymized etc) and the purpose of the query (who is the user). Different technologies are currently available to deal with authentication and authorization of API usage. Most APIs either use OAuth/OAuth2, user/password or API token via HTTP basic or digest authorization, or a special parameter included in each request. These all have advantages and disadvantages:

- OAuth and OAuth2 [2] are becoming the defacto standard for any API that primarily expects the API consumer to be a third-party application. It's secure, relatively consumer-friendly, and relatively easy to use. The downsides: implementations vary slightly across service providers, and it's a lot of overhead for something where you just want a single piece of data.
- User/password or special API token via HTTP basic or digest authorization is fast, doesn't require anything more than curl, and you can even make some request from a browser. The downside here is that it's not especially "friendly" to ask an end-user to go find an API token.

- User/password or API token as a URL parameter fortunately isn't very common. It's more confusing than basic authentication, has all of the same downsides, and no real benefit.

For the Smart Citizen Assistant API, most of the functionality provided by the Teradata Database can be reused and thus there is no immediate necessity to reinvent the wheel. Teradata supports:

- User-level security controls.
- Increased user authentication options.
- Support for security roles.
- Enterprise directory integration.
- Network traffic encryption.
- Auditing and monitoring controls.

5.5 Performance

The SCA will not provide real-time data nor will it provide big volumes of data. Thus low to medium performing devices and connections on the end-users side should suffice for operating SCA. The data is also simple in structure and not distributed. Although the API itself will provide no distribution services it could be used by applications combining data from different sources or as an entry point to a cloud based solution if the need arises. The SCA should provide data access fast enough to allow smooth and responsive app usage.

5.1 Quality of the API

As the smart citizen assistant API will have to support multiple representations of the resources (data), the API should allow content negotiation (eg. Accept headers), or differing URLs for different representations (eg. …?format=json). The API has to be stateless — this means the order in which the API methods are called should not determine the result. It is also important to build an API framework with some form of declarative security so that it's easy to assign and modify authorization rights on read and write access to resources.

Some clients will need bulk access to data, so these should be supported by allowing bulk operations (copy large datasets in one call). In such cases, it is also important to consider "pagination" which serves two big purposes in an API; it reduces the amount of unneeded data delivered to the client, and it reduces the amount of unneeded computation on application servers.

5.2 Mapping to Existing Wiener Netze Metering APP

The Wiener Netze Metering APP architecture consists of several components (see Figure 8):

- The phone app,
- A web server for static resources,
- An application server component for measurements,

- The corresponding MDMS,
- An application server component for standing data, which relies on an SAP system.

A web server for serving static resources such as icons, pictures, texts, or styles is necessary for a web application but might not be necessary for a mobile app. A database for user authentication and authorization is still necessary for both a web application and the SCA. In the SCA architecture the ASCR Teradata data center will partly provide this feature.

Standing data is not in scope of the SCA. This component is therefore ignored in the SCA architecture and in the API specification. The functions *authentication* and *authorization* which are also provided by this component must also be provided by the SCA architecture. These functions allow users to login to the system and then give access to a users data.

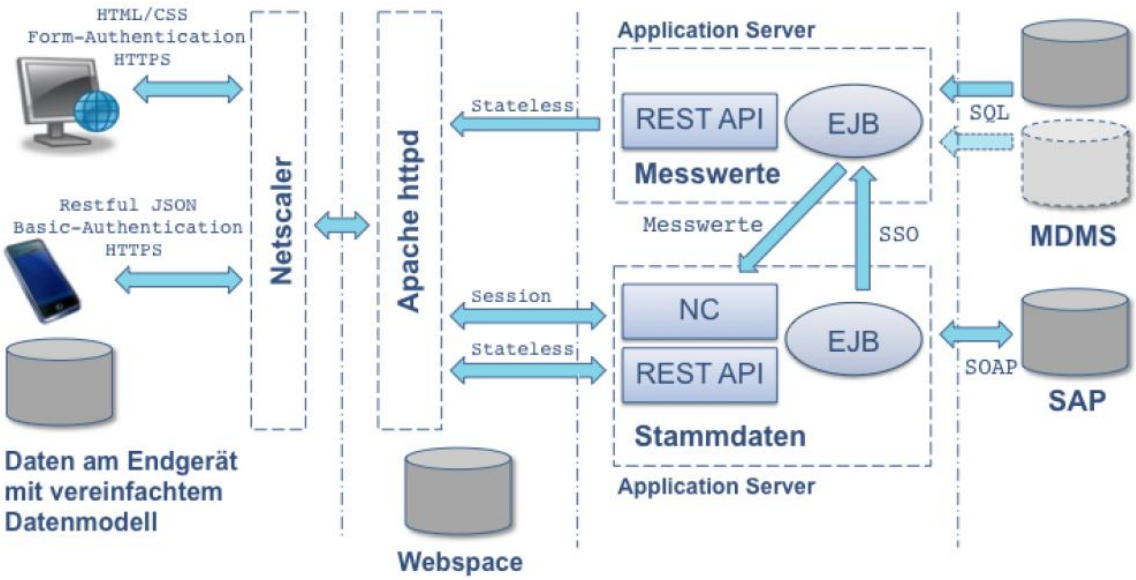

Figure 8 Architecture of the existing Metering APP of Wiener Netze [Source: Wiener Netze]

Instead of the application server component for the electricity usage measurements and the MDMS, the SCA architecture relies on the ASCR Teradata data center which in turn, imports also data of an MDMS.

6 Conclusions

To create an actual API it is necessary to somewhat constrain the possibilities. Most importantly to recognize bad API designs as early as possible. API design guidelines provide a repository of ideas and methods to help creating a good, usable API which is attractive to use for developers, and enables a satisfactory user experience.

Most API designs are over-constrained. Often APIs are designed for one specific use-case making it practically unusable for different use cases. By ignoring the requirements only specific to a single use-case, and concentrating on the common properties, API designers should strive for a simple API with a minimum number of assumptions regarding the use case.

Not everybody will be happy with the first attempts of API design. Thus API designers have to expect user feedback and plan ahead on how to address them (versioning is useful here again, but communication with the users of the API is important as well). After some time (a few years) an API can be considered stable and usually only minor changes are necessary.

6.1 Issues to be considered

Some of the research questions and issues to be addressed in this context are:

- How can the data access in different levels of granularity and data formats be supported by one API?
- What precautions are necessary to establish the Smart Citizen Assistant as a standard for city data?
- What measures are necessary to long-term maintenance and evolution of the data? What effects do these changes on existing applications?
- What aspects of data migration, data cleansing, data aggregation and data enrichment must be considered in the interface definition?
- What are the requirements for the interface arise in terms of data volume and response times of the issues typical end-user applications such as the "Energy Efficiency application"?

6.2 Examples of other (thinkable) innovative use cases for API usage

The SCA will contain a similar app as the Wiener Netze Metering APP. The new app will demonstrate the added value of accessing data through a data ware house (as opposed to one data source) by combining multiple sources of data and providing some new insights to the customer through the APP. **Examples** of such combination of data could include (not all of these will be necessarily implemented in the DEMO App):

- Combining weather data and smart metering data to analyze the effect of external temperature to energy consumption.
- Integrating smart metering and building data to provide insights to the building manager, as to how the self consumption of the building could be optimized.
- Integrating data available through open data sources (e.g., population) with aggregated smart metering data of a region to correlate the age of the customers and their energy consumption patterns.
- Comparison of aggregated energy usage data of different regions in order to find patterns of energy consumption and associating these to social status of people living in the different areas.

The foundations for technical prerequisites for such innovative use cases is laid in this project and the lessons learned during the process can be used to continuously improve the API, and avoid common pitfalls when integrating new sources of data or creating new APPS in the future.

7 Literaturverzeichnis

[1] Teradata, "Big Data: Teradata Unified Data Architecture in Action," November 2013. [Online]. Available: http://www.teradata.com/products-and-services/unified-data-architecture/. [Accessed 10 February 2014].

[2] D. E. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, 2012.

[3] S. Dustar, H. Gall und M. Hauswirth, Software-Architekturen für Verteilte Systeme, Berlin: Springer, 2003.

[4] J. Bloch, Effective Java. Second Edition, Addison Wesley, 2008.

[5] J. Tulach, Practical API Design, Apress, 2008.

[6] B. Mulloy, "Web API Design: Crafting Interfaces that Developers Love," [Online]. Available: https://apigee.com/. [Accessed 10 February 2014].