CityScripts: Unifying Web, IoT and Smart City Services in a Smart Citizen Workspace

Atta Badii[1]*, Davide Carboni[2], Antonio Pintus[2], Andrea Piras[2], Alberto Serra[2],
Marco Tiemann[1], and Nagarajan Viswanathan[1]
[1] *University of Reading*
Reading, U.K.
{atta.badii, m.tiemann, n.viswanathan}@reading.ac.uk
[2] *Information Society Department*
Pula (CA), Italy
{davide.carboni, antonio.pintus, andrea.piras, alserra}@crs4.it

Abstract

This article describes a programming experiment in which an entire city is the playground. Building on the foundation of the SmartSantander FIRE infrastructure, the CityScripts project proposes an Internet of Things scenario where sensors and actuators in a smart city have digital counterparts and can be used to compose mash-ups with Internet of Things devices, social networks and other online data sources and data sinks. This article describes both the general concept of CityScripts, a prototype implementation built on top of the SmartSantander FIRE infrastructure and a formative user evaluation carried out in order to determine the CityScripts user experience and in order to identify goals for the future development of the CityScripts platform.

Keywords: sensor networks, smart cities, Internet of Things, Internet of Services

1 Introduction

This article presents the goals, use cases, architecture, implementation and a formative user evaluation of CityScripts, an experimental system deployment within the large-scale European research project SmartSantander [1]. A system infrastructure and user interface for interacting with building blocks for a smart city have been implemented in the CityScripts project, which is part of the SmartSantander EU FP7 research project[1]. This article describes the achievements of the project at the end of its 12-month project duration[2].

The goal of the large-scale city-wide IoT installation at the heart of the SmartSantander project is to deploy a total of 20,000 sensors in the European cities of Belgrade, Guildford, Lübeck and Santander, with a deployment of 12,000 sensors in the city of Santander in particular [3]. The SmartSantander project encompasses a large variety of technologies and has the aim of providing researchers and the

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, volume: 4, number: 3, pp. 58-78

*Corresponding author: JJ Thomson Building, School of Systems Engineering, University of Reading, Whiteknights, RG6 6AY, United Kingdom, Tel: +44-118-378-7842, Web: `http://www.isr.reading.ac.uk/index.php/members/89-individuals/106-prof-atta-badii`

[1]The CityScripts project web site can be accessed at `http://cityscripts.crs4.it`; the SmartSantander project web site can be accessed at `http://www.smartsantander.eu`.

[2]This article is an extended version of the work originally presented at the 2013 International Workshop on Pervasive Internet of Things and Smart Cities (PITSaC'13), Barcelona, Spain, March 2013, titled "Scripting a Smart City. The CityScripts Experiment in Santander" [2]. This article has been comprehensively rewritten and extended relative to the workshop contribution. Updates and additional content have been added throughout all sections of the article and a new section describing the formative user evaluation carried out in CityScripts has been added.

general public with access to smart city infrastructures. Hence it is imperative to provide tools that enable all stakeholders, and in particular the citizens, to use the available infrastructure. In SmartSantander, using the infrastructure should encompass both using services enabled by the available infrastructure as consumers and using the available infrastructure as developers in order to develop new systems and functionalities. Tools that are provided for the latter should facilitate both simple application building and more complex development including statistical data logging and harvesting, stream data mining, inter-service communication, data conversion and exchange between infrastructure components and service orchestration for automation and service control purposes.

The CityScripts project has developed such tools for open urban service development. To this end, the project proposes an Internet of Things scenario in which sensors and actuators in a smart city have digital counterparts which can be used to connect services and to compose mash-ups, which are compositions of components that integrate sensor data with Internet of Things devices, social networks and other online data sources and data sinks. An important aspect of the project is the development of a personal workspace in which each user can compose public data from city sensors with public online data sources and with personal data from her/his personal devices and services. CityScripts users can interact with physical and virtual sensors and actuators using a unified approach; users have control over the visibility of sensors and actuators using a simple rights model and can connect with their friends and contacts, which allows them to share physical or virtual devices with others.

The two main goals of the CityScripts project are 1) to investigate architectural and practical issues, solutions and opportunities for integrating smart city sensors and actuators into a personal IoT workspace, and 2) to gather user feedback and usage data by exposing the system to users. The target audience of the CityScripts experiment are persons with practical interest in using the services provided by the platform as part of their daily living or working activities; envisioned usage scenarios are expected to be non-trivial. This definition includes early adopter "geeks", researchers, programmers and makers. At the stage of development discussed in this article, the system is not expected to be at a development maturity level where it is instantly usable by an average Internet user; insights from work with early adopters are gathered in order to progress development towards broader usability and more useful and advanced features.

In order to achieve the objectives of the CityScripts experiment, Cloud computing and service-oriented architecture concepts are used. In the context of CityScripts, these architectural approaches allow Internet of Things applications to achieve a higher level of interoperability and usability. Cloud-deployed services can interact via multiple protocols, work with a variety of data formats and can act as brokers that combine services in a logical space. Importantly for CityScripts, they can act as proxies for data originating from data sources with very heterogeneous characteristics ranging from sensor nodes to online services (such as social networks, news feeds or Web Services), and can relay data to consumer services. Moreover, they can adapt data formats where required.

The CityScripts platform is based on the Paraimpu platform, which has been developed by CRS4 as part of earlier research activities. Paraimpu is a social tool focused on the Internet of Things; it is available to the interested public in a closed alpha online version [4]. The CityScripts platform inherits the main architecture and infrastructure features of Paraimpu: fundamentally, the system is a Web application with which the end user interacts using a browser-based online workspace. In the workspace (depicted in Figure 1), users can add, edit and delete digital instantiations of physical objects ("things") and/or social network or Web Service objects and other applications, such as Facebook or Twitter accounts, which can be selected from a palette of available object types.

The objects available in the palette (physical as well as other types of objects) are grouped into two main categories: sensors and actuators. An object belonging to the sensor category is an object capable of producing data, for instance a temperature sensor or a set of sensors deployed in a city, while an object belonging to the actuator category is an object able to consume data, generally in order to execute an

Figure 1: CityScripts Web workspace. The workspace provides the end user with a palette of classes of "things" that can be connected through the system, including physical objects, social networks and Web Services.

action, and could for instance be a stepper motor, a lamp light switch or a user's Facebook messaging "wall".

Users can establish connections between sensors and actuators. These connections are logical data flows between the two objects. They can be configured and customised using JavaScript code fragments in order to filter or transform data as it flows from a sensor to an actuator. This feature allows the overall system to provide great flexibility and power in connecting quite heterogeneous types of objects, including physical and software-based "virtual" objects.

Another facet of the CityScripts workspace focuses on a social sharing aspect: each object created in a workspace can be shared with other users, so that users can let friends or other persons use their devices or services created in their personal workspaces in order to build personalised applications that use their friends' devices and services.

CityScripts provides all of the features described above and adds additional important functionalities relevant to smart city device integration and experimentation:

- A new sensor class, the Abstract Region Virtual Sensor, collects and aggregates data measured by sensors in a specified geographic area of a city and of a specified type. Aggregation is computed according to user-selected aggregation functions and data output is provided in terms of aggregated data;

- A second new sensor class, the Abstract Region Geographic Selection Sensor, collects data measured by sensors of a specified type and from a specified geographic area of a city. No data

aggregation is performed and data flows directly from each sensor contained in the geographic bounding box defined by users;

- Scripting in JavaScript, for which the workspace provides a scripting environment that enables the creation and cloud-based execution of logic working on real object and services through JavaScript code snippets.

CityScripts integrates a prototype context-aware reasoning engine and data representation framework developed previously as part of the Hydra Middleware EU FP6 IoT middleware project, which is publicly available as part of the LinkSmart open source IoT middleware [5]. The original LinkSmart Context Awareness Framework has been modified and has been integrated with the Paraimpu platform and the SmartSantander sensor framework in order to provide functionalities to end users that allow them to reason with contextual and situational data when accessing physical sensors, virtual sensors and other suitable CityScripts-enabled data sources.

The adapted component that is used in CityScripts has been named CityScripts Situation Awareness Framework (SAF) in order to distinguish it clearly from the LinkSmart Context Awareness Framework. It supports context modelling and context processing using a rule engine in an object-oriented architecture. It provides extended rule-based reasoning functionalities through the integration of the well-known JBoss Drools rule engine [6] with a custom tree-based hierarchical data storage and access model. The Framework supports Complex Event Processing (CEP), which can generally be defined as an event processing concept that deals with processing multiple events with the goal of identifying relevant or meaningful events within large event clusters or streams. Complex Event Processing can in this context be used to detect complex event patterns and correlations between events. It can make use of event hierarchies to support abstraction and to represent relationships between events such as membership, temporal ordering and causality.

The provisioning of the Situation Awareness Framework is expected to benefit CityScripts developers by providing three new opportunities: first, to enable developers who are familiar with the JBoss Drools rule language to implement complex rule-driven scenarios using the SmartSantander FIRE testbed; second, to experiment with and to improve the application of state-of-the-art contextual rule-based reasoning systems [7] when deployed on "city-scale" environments with their specific requirements originating from the number of sensors used and data points involved as well as to the behaviour and reliability of the gathered data; third, to investigate potential use cases for CEP in smart city environments with large numbers of IoT sensor data sources distributed over a large urban area.

2 Use Cases

The following two scenarios describe typical applications within the scope of the CityScripts project: one is a consumer scenario, where the end user manages a personal workspace to trigger actions on social networks when some conditions concerning incoming city sensor data are met. The second scenario is business-oriented and depicts how a small business can collect data, make inferences and take decisions in response to traffic data inferred from environmental sensor information alone. The core idea of that scenario is to provide the ability for businesses (and consumers) to become active actors in and beneficiaries of smart city infrastructures in order to provide better services. These two use case scenarios show that both consumers of a service, in this example a restaurant manager, and business developers – in the second example presented – can benefit from CityScripts if suitable tools are provided for them.

2.1 Consumer Scenario

A restaurant manager wants to keep customers who follow her restaurant on Facebook informed about the nearby parking capacity in real time. By providing this service, the restaurant manager hopes to keep potential customers updated who may decide to come from their office to her restaurant during the rush hour about the most suitable transport modality for a stress-free visit. Accessing raw data as provided by a smart city environment would not address her need, because data from parking slot sensors are not designed to trigger events or actions in the way the restaurant manager would like to use the available data in order to provide information. The restaurant manager needs a tool that allows her to integrate data from several smart city data sources, process them, and push the processed data to the online social network destination representing the restaurant.

The manager logs into her CityScripts online workspace and selects a sensor that lets her select a geographical area and the required type of sensor from the sensors available from the smart city systems, naming it "parkingSlots". She then creates an instance of the restaurant Facebook account connection as an actuator to which the CityScripts system can send messages, naming it "facebookWall". She then creates a connection from the "parkingSlots" sensor to the "facebookWall" actuator. The configuration options available in the sensor, actuator and connection allow her to ensure that the correct information is sent from the sensor and arrives on the restaurant Facebook "wall" as an appropriately worded message that still contains the relevant information about available parking slots. Now her contacts can find information about the number of available parking slots directly on their mobile device or PC and can decide which transport modality to use when visiting the restaurant.

2.2 Business Scenario

A delivery agency wants to use available smart city data to provide van drivers with up-to-date information on the urban traffic situation, including statistics on traffic flows over time. Since this requires the development of a software product, the agency tasks a software house with developing the necessary software. The software house decides to use the CityScripts platform as a middleware between the smart city source data and a custom-developed application for the small smartphone-style devices provided for the delivery agency van drivers.

The software house uses the generic sensor available in CityScripts to input the location of delivery vans into CityScripts and uses the available smart city sensors to gather information on traffic density. Scripting and reasoned rule sets are used to process and gather these data and format the resulting output data suitably for presentation on the client devices. At the end of the implementation, the delivery agency has access to a powerful traffic display and analysis tool that can be customised for each delivery van driver and can be extended easily as new sources of information, such as automated information on road closures, is made available.

3 Architecture

3.1 Overview

The CityScripts architecture, depicted in Figure 2, consists of a Workspace server, a SAF server and connectors that integrate various data sources as sensors and data sinks as actuators that receive and use provided data. CityScripts connects to the SmartSantander Ubiquitous Sensor Network (USN) component in order to connect to sensors deployed throughout the city of Santander. A publish-subscribe mechanism is used to receive data event-driven, thus avoiding the need for continuous polling for data.

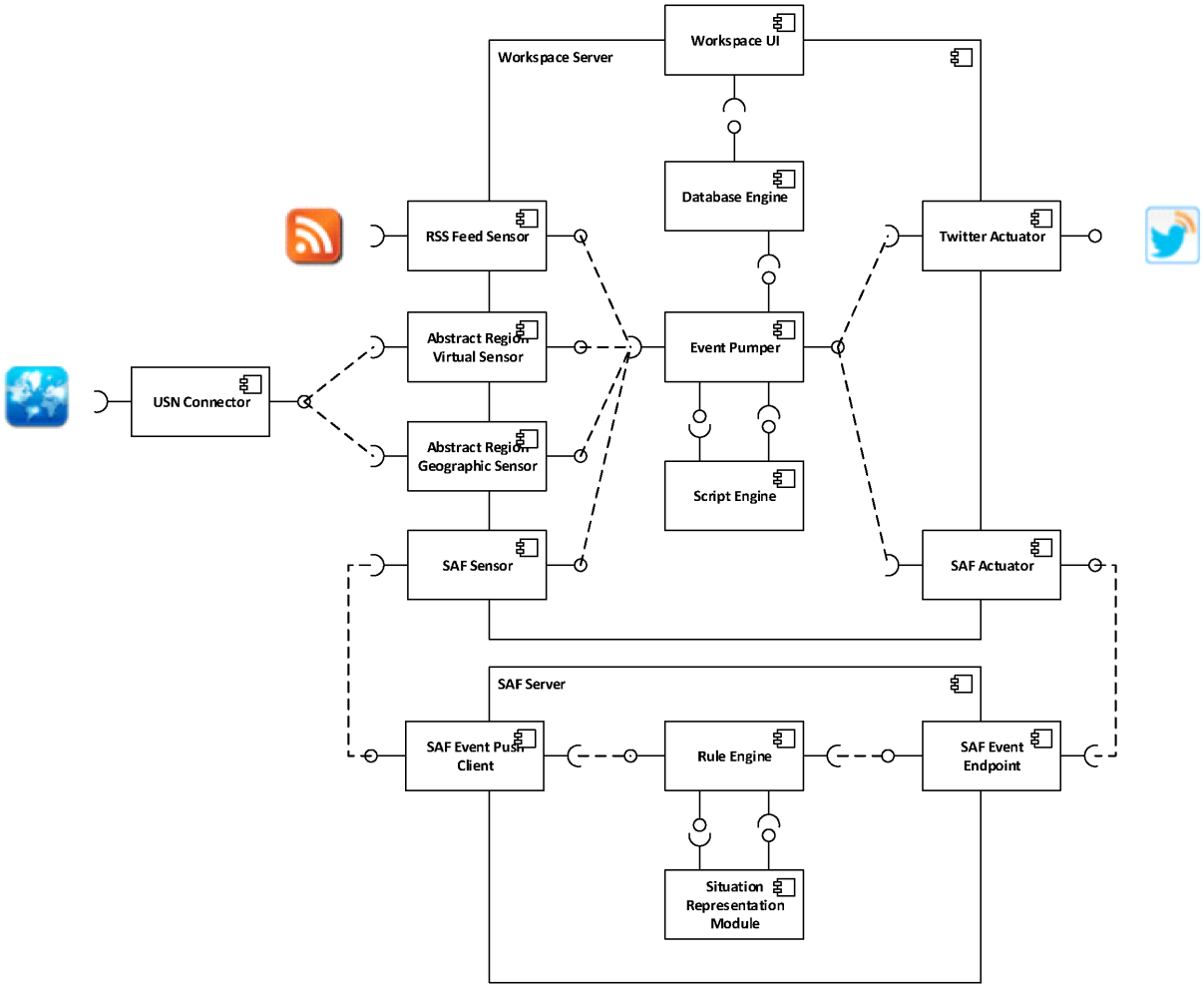


Figure 2: CityScripts architecture overview. This overview depicts they main components of the system and examples for sensor and actuator connectors; not all connectors are depicted and a number of functionalities have been omitted from the diagram for clarity; coloured icons depict external RSS and SmartSantander data sources and a Twitter actuator.

As can be seen in the figure, connector components are used to connect sensor and actuator objects to the CityScripts system. Both generic connectors (not depicted in the figure) and customised connectors are available in CityScripts. The workspace and SAF system components are implemented in separated server environments (see below for further details). The following subsections elaborate on specific components and functionalities of the CityScripts system.

3.2 Personal Social Workspace

Commonly, smart city systems and services are envisioned as hidden, “invisible” systems; for instance, the citizen is not usually expected to directly interact with or access sensor data from smart city sensors. Instead, the citizen tends to be described as a consumer of services which are enabled by smart city environments that involve sensors and actuators of various types. In CityScripts, it is envisioned that a citizen may have the desire to interact with smart city infrastructures as well as other IoT-enabled environments in a much more direct way, for instance by directly viewing and analysing provided data

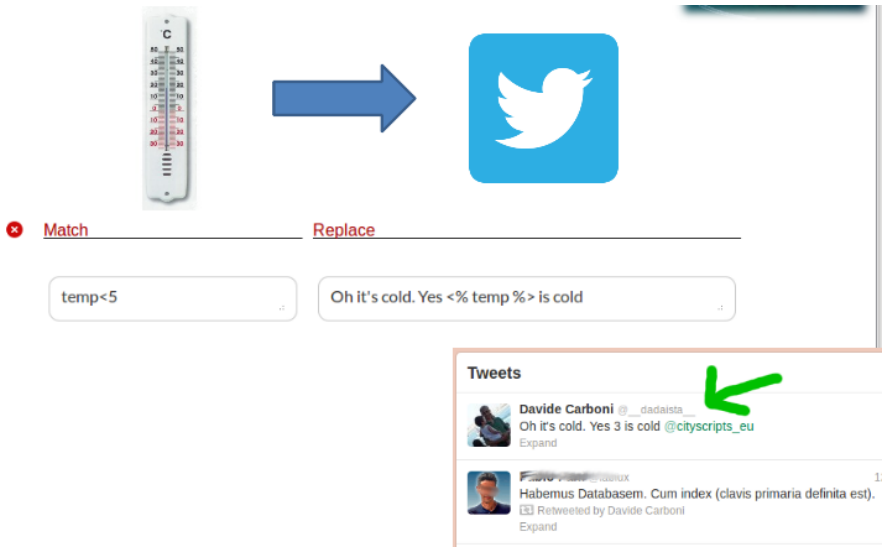


Figure 3: An example of a connection between a temperature sensor and a social network account. While this example shows only a single match/replace statement, a list of numerous statements can be defined on a connection, and the first match condition that is met will be executed to transform data.

and in particular by creating services of her/his own.

Systems such as COSM [8] aim to facilitate such interests by providing tools for data collection and display, for instance in order to gather physical sensor data to display trends over time. Such functionalities can be useful but can also be considered to be merely a minimum relative to what could be provided. The CityScripts workspace aims to provide the ability to integrate city data with "If This Then That" [9] logic, simultaneously integrating city data with personal devices and other Internet services. Figure 3 illustrates an example of a connection between a sensor and an actuator and a match/replace statement to conditionally modify the data flow. As already mentioned, the social aspect of the platform allows users to share sensor objects with other users; in CityScripts, a user's Twitter contacts are the social circle of interest. There is no technical constraint to only being able to use Twitter as a social network – using Twitter in CityScripts has been chosen just to avoid re-implementing interfaces to additional social network platforms from scratch during the reasonably short project.

3.3 Abstract Regions and Virtual Sensing

Using one or a few specific urban sensors can be the preferred option for an end user who would like to get specific information from a single specified data source. In other cases, the user may be more likely to be interested in data flows from selected contexts in a city, such as a particular geographical area. When providing information on available parking spaces to customers for example, it may not be very useful for the user to know whether a specific car parking slot is free, but it would be more beneficial to know how many car slots are available within the geographically definable boundaries of a surrounding area. The Abstract Region Widget (Figure 4) is a component that lets the user select a region on a map screen and generates a geographically bound data flow.

In the context of CityScripts, the term Abstract Region is used to denote a virtual sensor whose data are the combination (linear or according to other rules) of data which are coming from a given geographical bounding box. Once created, an Abstract Region can be used and composed in the user workspace exactly like any other sensor or service (Figure 5). An Abstract Region can thus be used exactly like a single physical sensor representation would be used.

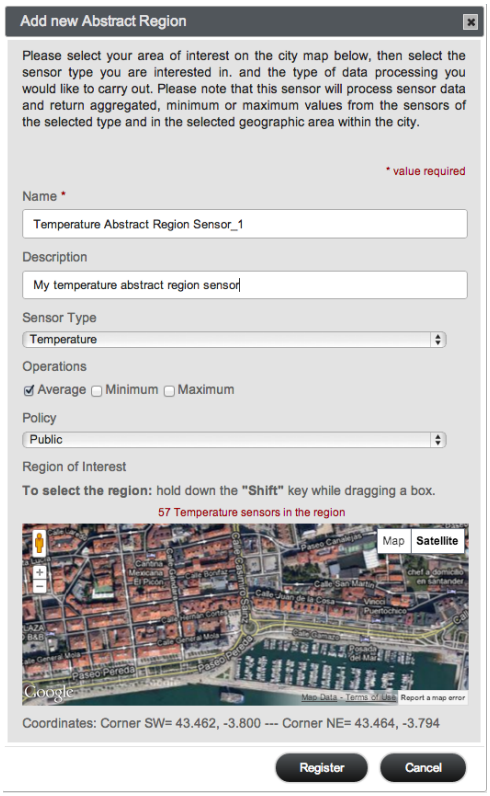


Figure 4: Abstract Region Widget for selecting sensors in a geographical region.

3.4 Scripting

The objective for the integration of scripting functionalities in CityScripts is to enable more complex logic than can be facilitated by direct sensor-actuator connections. For instance, logic such as

onDataEvent from sensor X:
    take values of sensors A,B,C ;
    compute y=f(X,A) and z=g(X,B,C) ;
    send y to actuator D and z to actuator E

that processes input from multiple sensors and addresses multiple actuators cannot be implemented in CityScripts using a single connection from a sensor to an actuator, but it can easily be implemented with a small script running in the scripting engine that has been integrated as part of CityScripts. To fully understand the limitations imposed by the one-to-one connection between sensor and action, consider the following practical example of a thermostat logic. A thermostat can be considered as an ON-OFF logic where a) if the temperature is above a threshold T, the heating is turned OFF, and b) if the temperature is below T, the heating is turned ON. If one attempts to write the logic in a sensor-actuator connection inside CityScripts, there would need to be two mappings with two different match/replace expressions:

match: temperature >T
replace: OFF

match: temperature <= T

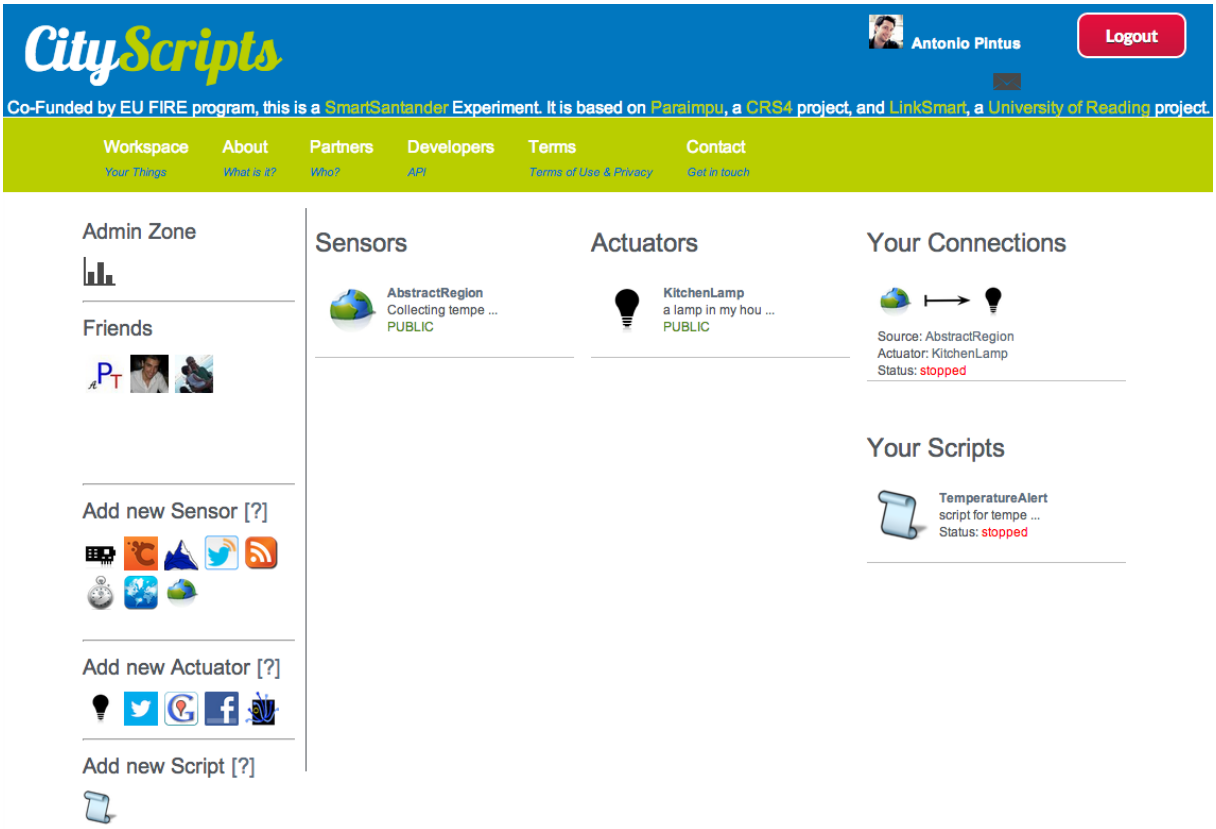


Figure 5: A screen shot of the user workspace in the CityScripts user interface. The figure depicts the user's social circle, a sensor, an actuator, a connection between the sensor and actuator, and a script instance.

replace: ON

But unfortunately this data mapping from sensor to actuator would not correctly implement the thermostat logic. In fact, if one assumes the threshold T=0 and assumes the sensor produces the values: -1,+1,+2 etc., the solution with two mappings will obtain an output of ON,OFF,OFF, while the goal is to obtain OFF only when the temperature moves from -1 to +1 and not when it moves from +1 to +2. This means not only must the actual value of a sensor be observed but also the previous value. The logic should hence be rewritten as follows:

If old_temperature $<$T and new_temperature $>$T then OFF
If old_temperature $>$T and new_temperature $<$T then ON

In order to enable logic based not only on current values but also on past values, the scripting engine is equipped with built-in functions that are directly accessible as JavaScript functions in the scope of every script. The available functions include functions that allow users to retrieve metadata of sensor and actuator instances, to read single and multiple stored values from sensor instances and to send values to actuators. At the implementation level, the script manager is the component that allows the user to manage the Create-Read-Update-Delete (CRUD) lifecycle of scripts. It consists of graphical user interface elements integrated into the CityScripts workspace to create, remove, update and delete scripts. A

CityScripts script has the following structure:

1. The script: the script to be executed, written in JavaScript;

2. Script dependencies: the dependencies that the script requires to be satisfied in order to operate; in the initial version, only references to a pre-defined set of built-in functions are permitted in CityScripts;

3. Script trigger: the sensor that causes the script to be executed; A script is executed every time new data is sent to the script from a trigger sensor, also including sensors such as timer-based sensors that dispatch messages in regular intervals.

The CityScripts system version considered in this article provides users with a console window and a basic testing functionality that can be used for debugging purposes. The scripting engine is based on the Rhino project, a robust JavaScript interpreter written in Java. A Java servlet running in a Jetty web application server carries out the JavaScript evaluation and communicates via HTTP. To prevent infinite loops and malicious code in scripts pre-emptive killing is adopted: each script has a limited time window (e.g. 500ms) available to evaluate and return - if it takes more time the script is killed and not simply suspended.

3.5 Situation Awareness Framework

While CityScripts scripting functionalities empower users who have suitable JavaScript development experience to implement complex procedural processing sequences, the scripting system does not provide users with specialised libraries of functionalities for processing events, nor does the scripting language itself provide optimisation methods for handling very large amounts of event data efficiently.

The CityScripts Situation Awareness Framework has been integrated in order to provide such features to users of the CityScripts system. The Situation Awareness Framework integrates the JBoss Drools rule engine into CityScripts and enables developers to formulate event processing rule sets in the expressive rule expression language of the Drools rule engine. As using the Drools rule expression language requires some familiarity with rule language syntax and fundamental knowledge of computer science topics, it is envisioned that the Situation Awareness Framework would predominantly be used by enthusiast hobby developers and by commercial users of a CityScripts system.

A generic tree-based and object-oriented data structure which can be used to store and access instance and context data in a hierarchical fashion has been implemented; the reasoning engine can read from and write to this data structure in order to retrieve stored data and store any results of reasoning processes that are to be stored in system memory. The tree-based hierarchical data representation model should enable developers to easily store and retrieve data in a wide range of application scenarios.

The Situation Awareness Framework is implemented as a separate standalone server. It connects to the CityScripts workspace server through custom sensor and activator implementations. Users can connect any sensor to a Situation Awareness Framework actuator instance, so that sensor data is sent to the Situation Awareness Framework using the standard connection functionalities. The output of the Situation Awareness Framework is then provided through a custom Situation Awareness Framework sensor that is automatically added to a workspace when a Situation Awareness Framework actuator is created. Figure 6 depicts the data flow of this integration; Figure 2 shows the integration of the Situation Awareness Framework into the overall system at component level.

Developers can define event-driven rules and sets of rules that work directly on received input data and also on stored contextual data. These could be simply stored “raw” data or also aggregated data or data that has been processed otherwise. Consider for instance the following rules as a simple example

for rules in a smart city application (described in a simplified format due to space restrictions):

Rule "LightIntensityDeviation"
When
(InMessage.valueType == "lightintensity") && (deviation: InMessage.value -
LightMemory.get(InMesssage.location, LightMemory.timeOfYear, LightMemory.timeOfDay)
>LightGoals.maxIntensityVariation)
Then
Message (InMessage.location, deviation);

Rule "NeighborLights"
When
InMessage.valueType == "lightidentifier"
Then
Message(LightMap.getNeighbors(InMessage.value, LightMap.defaultRadius));

Rule "LightIntensityCorrection"
When
(InMessage.valueType == "lightintensity") && (deviation: InMessage.value -
LightMemory.get(InMesssage.location, LightMemory.timeOfYear, LightMemory.timeOfDay)
<LightGoals.maxIntensityVariation)
Then
Message(InMessage.location, "corrected");

This example of a rule set could be employed together with the situational data memory of the Situation Awareness Framework and the CityScripts sensing/actuating and scripting system in order to a) identify a technical failure of a city light given the its location, its usual brightness target, the time of year and the time of day via an attached light intensity sensor (rule "LightIntensityDeviation"), b) identify city lights close to the defective light (rule "NeighborLights"), and c) incrementally increase the light intensity of the neighboring city lights until they compensate for the loss of light from the broken city light (rule "LightIntensityCorrection") either for a specific "spot" or an overall area covered by a group of city lights.

The example rule set uses a "Message" function to send data to the CityScripts user workspace. While rule evaluation and direct messaging are of course necessary for the SAF system to function in the context of CityScripts, the Situation Awareness Framework supports additional actions, including to store data in the context memory of the Situation Awareness Module, from which stored data can also be retrieved for rule evaluation. These functionalities are important for gathering contextual data for situation-aware processing, because they enable the acquisition of data for purposes other than rule evaluation.

The Situation Awareness Framework is implemented using CityScripts sensor/actuator interface specifications, the components can easily be integrated in particular into the scripting environment. Following this approach, the specialised contextual rule processing environment of the Situation Awareness Framework can easily be combined with the generic and flexible scripting capabilities of CityScripts scripting in order to produce powerful advanced applications that take advantage of the respective advantages of the specialized rule processing and scripting engines.

As described earlier and as illustrated in Figure 2, the user workspace and the Situation Awareness Framework are loosely coupled and are executed in separate server environments. This can be both beneficial and problematic depending on the usage context. The separation of event flow processing

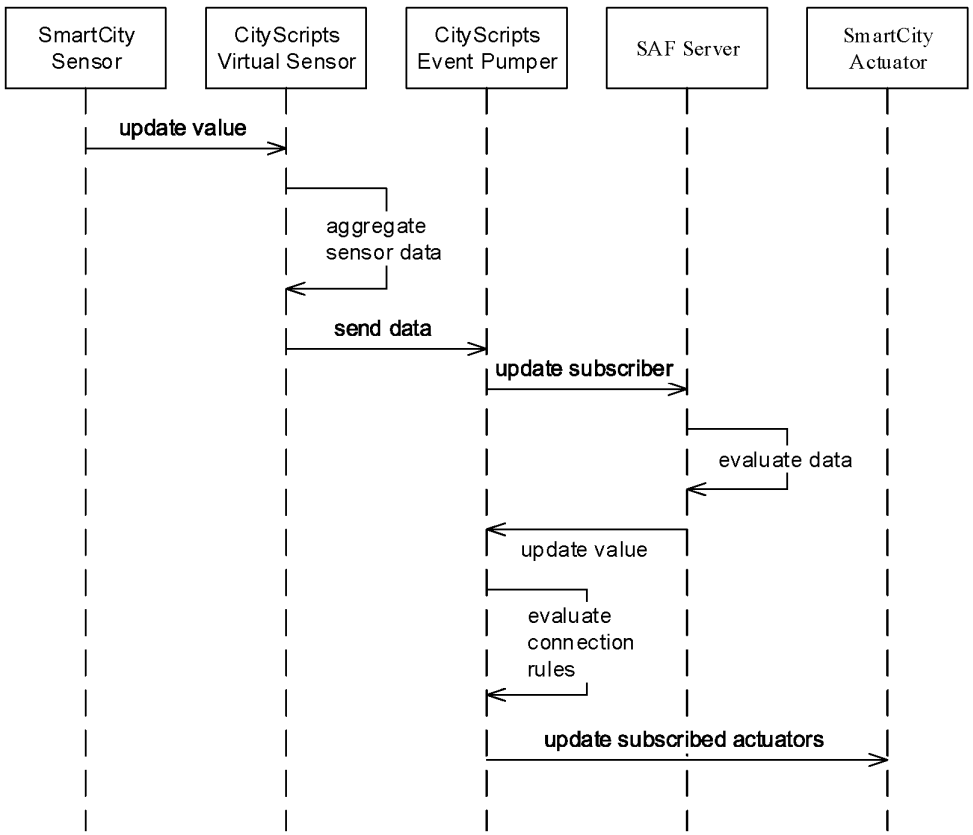


Figure 6: Integration of SAF as part of the CityScripts sensor-actuator infrastructure.

and complex event processing facilitates the implementation to use mature specialised rule processing engines with implementations of advanced algorithms (such as the Rete algorithms) and proven performance characteristics and an expressive language for specifying rules and rule sets for the engine, while at the same time retaining the ability to use generic scripting to implement other functionalities. It must however be ensured that both components are suitably scalable and performance-optimised and that performance benefits gained by employing specialist engines are not nullified by communication overheads. In addition, developers may need to familiarise themselves with both scripting and rule languages in order to utilise the system as described above.

From the perspective of scalability, the ability to scale user workspace and Situation Awareness Framework implementations separately allows the CityScripts system to support large-scale rule processing applications such as the generation of citywide near-real-time traffic maps using data from city sensors only or from city and vehicle sensors in combination. In scenarios such as this, the Situation Awareness Framework can be employed as a specialised solution available to a smaller number of "power users" intending to process large amounts of incoming data with large and complex rule sets, while the CityScripts system without the Situation Awareness Framework can be employed in less demanding situations; in commercial applications, this could lead to different levels of service provisioning that may be billed differently depending on data usage and processing performance.

4 Beyond the State of the Art

CityScripts encompasses a number of technologies and functionalities where Internet of Things (IoT) technologies meet the concept of smart cities. Smart cities are defined along multiple dimensions of description including economy, mobility and governance. Smart city technology plays an important role as a necessary enabler for smart cities [10] based on three main pillars: IoT, the Internet of Services (IoS) and the Internet of People (IoP), in which people are "part of ubiquitous intelligent networks having the potential to seamlessly connect, interact and exchange information about themselves and their social context and environment". CityScripts integrates all of these pillars in a single unified workspace: integrating smart objects and sensors, integrating online services and enabling social interaction and sharing.

While in the past it has been quite common for a device to operate in isolation and to execute custom or proprietary operating logic in isolation, networked decision-making and management of billions of devices within the cloud will become a core property of the Internet of Things, given the availability of an increasingly pervasive connectivity. A number of projects and initiatives that are primarily concerned with Internet of Things topics have focused on enabling the necessary communication infrastructure with new classes of systems and devices that can be part of IoT environments and which can exhibit very heterogeneous and wide-ranging technical capabilities and interfaces, hardware and power limitations and other properties, thus necessitating the creation of suitable new communication protocols in order to integrate them into IoT environments. CityScripts does not primarily focus on the technical communication level, for which the project relies on available interfaces and on the work carried out on hardware, protocols and interfaces within the SmartSantander project.

The CityScripts connector architecture integrates services and devices at a level of abstraction that allows the workspace to be generally agnostic of the specifics of the hard- and software employed, as long as minimum requirements concerning messaging and message formats employed are satisfied; the focus of CityScripts research and experimentation is put on the application level. This approach has gained acceptance for instance for smart things in a general Internet of Things context [10] that can become fully integrated into the Web when technologies and models commonly used for "traditional" Web content and services are reused and adapted for IoT devices.

CityScripts extends the scope of sensors and actuators that can be used in the CityScripts workspace beyond sensors and actuators available through the SmartSantander smart city USN connection to also include Web online services, social networks and personal devices. Although tools to manage data from Internet services already exist (for instance Yahoo! Pipes [11] or more recently IFTTT [9]), and although some allow integration with physical objects, this is the first time that such a service also exposes sensors and data of an entire city in addition to other devices and services.

CityScripts uses the concept of Abstract Regions in order to interact with smart city sensor data provided by the SmartSantander infrastructure. It is important to highlight the difference of this concept from the concept of Abstract Regions [12], where Abstract Regions are described as a means to capture a range of common idioms in sensor network programming. The Abstract Region concept in CityScripts focuses on a subset of the features identified in Welsh and Mainland [12], and in particular on reduction and enumeration, as these were most relevant within the scope of the project. The Abstract Region concept in CityScripts was used as a practical solution to the problem of accessing smart city infrastructure and not in order to further or develop a different programming paradigm for sensor networks. Nevertheless, the benefit of summarising the output generated by a set of sensors in an Abstract Region with a single represented entity is retained in CityScripts.

CityScripts provides social connectivity through the ability to share both output (for instance through social networks) and tools of the system (e.g. personal sensors shared with friends). Users can use the tool for personal purposes but also discover what other people in their social circles are sharing and

they can create services with data and objects of a friend. In Guinard et al [10], the authors propose an architecture for sharing devices based on a central module which keeps access credentials and maintains an access control list to decide which functionalities of a device are exposed to whom. The CityScripts approach is equivalent to a subset of the above, but starts from the concept of bookmarking, which is explained more in-depth in Pintus et al [13].

5 User Evaluation

As part of the project, a small-scale user evaluation with users representing the target audience of CityScripts at the stage of development reported on in this paper was carried out. The user evaluation addressed two main questions: 1. Is the CityScripts system and the functionality it provides accepted and appreciated by users? 2. What are the benefits and points for improvement for CityScripts in terms of system usability?

In order to address these questions, a focused evaluation approach with a small number of participants and in-depth analysis of their experience while using the system and their attitude towards the system before, during and after having used the system was carried out. Eight participants (7 male, 1 female) with a research background and several years of programming experience were selected as participants that could suitably represent the targeted audience of technically proficient early adopters.

The evaluation itself adopts the UI-REF evaluation approach [14] and includes assessments before using a system, while using a system and after having used a system. UI-REF was also employed in order to analyse evaluation outcomes and derive actionable and prioritised recommendations for the partners involved in the CityScripts project (not reported on in detail here).

This article provides a brief summary report of user evaluation findings. Additional information on the user evaluation, additional quantitative data, details on user responses while "thinking aloud" and during semi-structured interview sessions and details on the questionnaire instruments used can be found in the CityScripts project deliverable D1.3 [15].

5.1 Instruments, Measures and Procedure

In order to evaluate the CityScripts system, participants were invited to an evaluation session in a designated room for an evaluation session lasting between 45 and 90 minutes. During the session, they were asked to complete six tasks using the CityScripts system on a desktop computer (see Figure 7); the tasks were structured in the form of a tutorial of increasing difficulty. At the beginning and end of the session and between tasks, the experimenter conducted brief semi-structured interviews with the individual participants. A pilot experiment was used to test and revise the experimental setup before carrying out the experiment sessions described in this article.

A combination of evaluation instruments was employed for the evaluation. The procedure combined standard questionnaires with the "thinking aloud" technique, in which participants are asked to comment on their impressions and activities while using a system, and semi-structured interviews between evaluation tasks and at the end of each evaluation session. In addition, participants were recorded using a camera and microphone during the evaluation sessions, and their screen activities were recorded on the evaluation computer using suitable screen recording software.

For the questionnaire, an established instrument for measuring technology acceptance was employed. The selected e-TAM questionnaire instrument [16][17] is an extension of the well-known TAM Technology Acceptance Model questionnaire instrument. e-TAM extends this by items for measuring the Perceived Enjoyability of the system under consideration, which is considered to be important in the context of CityScripts in order to retain early adopter users of the system. The measured constructs of the orig-

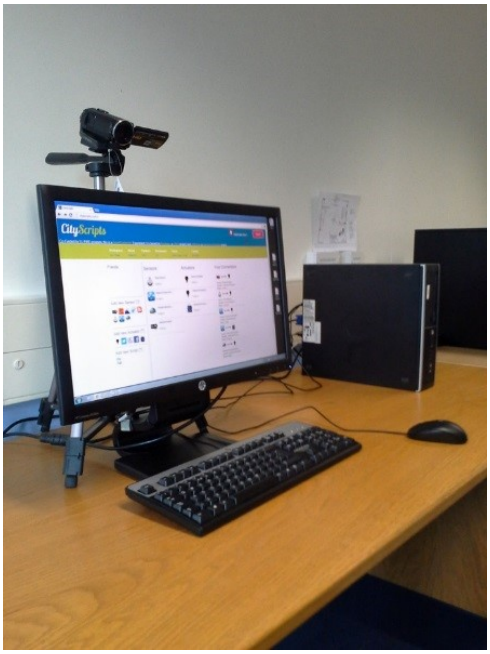


Figure 7: Screen setup for user evaluation.

inal TAM questionnaire, Perceived Usefulness, Perceived Ease of Use, Intention to Use and Perceived Quality, are part of the extended e-TAM measure and were also elicited. In a pre-test questionnaire, participants were presented with a brief written description of the CityScripts system and asked to complete the Perceived Usefulness questionnaire items of the e-TAM based on the provided information. The full questionnaire was administered post-test. Results of the questionnaire analysis in this small-scale experiment should be reviewed cautiously due to the small number of participants and should be considered as indicative; in order not to mislead the reader, statistical measures which would commonly be reported for data from larger user populations are not reported in this instance.

Thinking aloud and semi-structured interviewing were employed during the evaluation sessions and a final semi-structured interview session was carried out at the end of each evaluation session. Questions in the semi-structured interview were concerned with the satisfaction with completing a task, specific good and/or bad elements encountered while completing a task and suggestions for changes in order to better achieve a task; the experimenter was free to ask additional questions regarding observed usage behaviour by the evaluation participant while completing a particular task.

5.2 Results

Eight participants completed the evaluation sessions: 7 male, 1 female, aged between 20 – 35 years, with 2 to 12 years of programming experience according to self-report (mean 5.88 years, std. dev. 3.451), participated in the evaluation. All participants had knowledge in using one or more of the programming languages C, C++, C# and/or Java.

Figure 8 and Figure 9 show the participant responses to questionnaire items intended to measure the Perceived Usefulness of the CityScripts system before and after having used the system. The depicted count of responses indicates that overall, participants perceived the system to be at least as useful as expected after having read a brief description of the system, and in several instances more useful.

Responses concerning the Perceived Usefulness of the CityScripts system are mirrored by participant responses during semi-structured interview sessions. CityScripts was generally considered to be useful

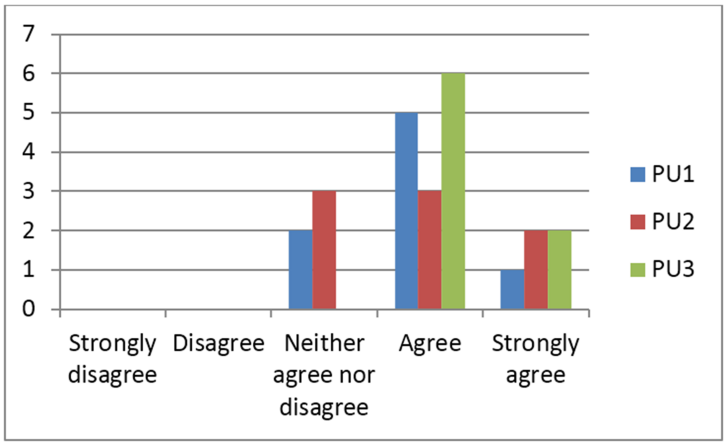


Figure 8: Perceived Usefulness Pre-Test response distribution.

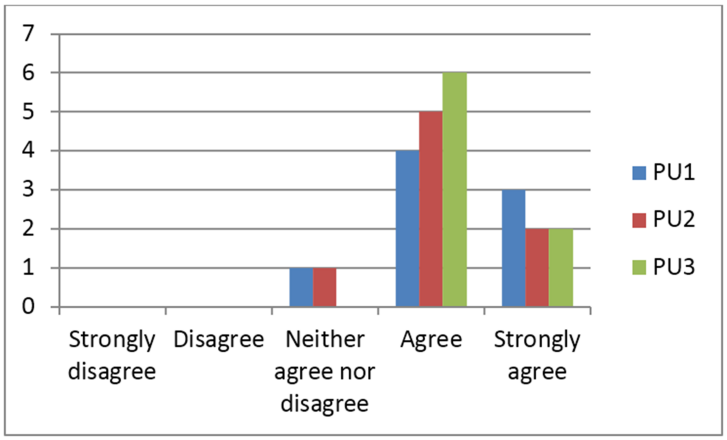


Figure 9: Perceived Usefulness Post-Test response distribution.

for carrying out the tasks which the participants were asked to carry out as part of the evaluation, and the system was considered to be useful for integrating Internet of Things and Smart City devices into a service-based infrastructure by all of the participants.

The overall assessment of the Perceived Ease of Use of the system was not as positive as for the Perceived Usefulness. Through thinking aloud and through semi-structured interviews, it was found that participants perceive the usability of different components and functionalities exposed by the system differently. The overall approach of instantiating and connecting sensors and actuators of any kind using a web interface was evaluated positively by participants and considered to be simple, useful and intuitive. Advanced functionalities, in particular those involving scripting and the use of the Situation Awareness Framework, were evaluated as being less easy to use than the more guided functions concerned with creating sensor – actuator connections.

Perceived Enjoyability was evaluated neutrally by participants. Participants did not strongly consider the CityScripts system to be fun to use. During semi-structured interviews participants provided further information; several participants stated that they would find the system to be more fun to use if it provided more options for available sensors and actuators that they could use with devices and services that they were already using in their daily life. Generally, participants appeared to consider the CityScripts system to be of utilitarian value rather than an enjoyable experience for “play”.

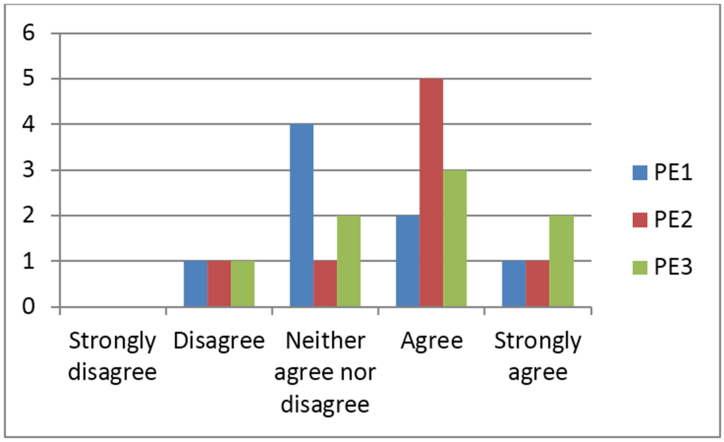


Figure 10: Perceived Ease of Use response distribution.

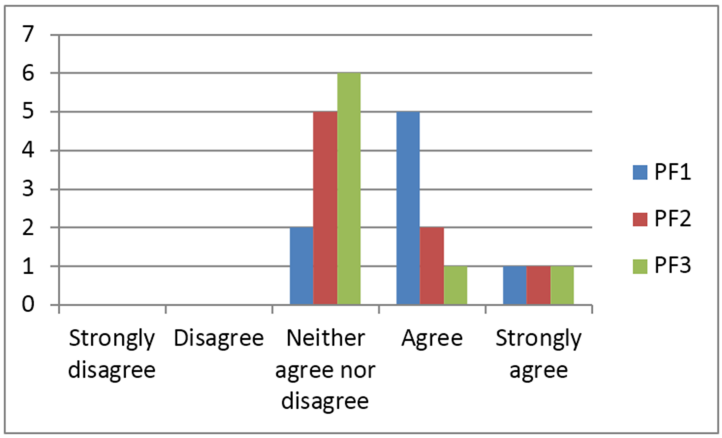


Figure 11: Perceived Enjoyability response distribution.

When asked whether they would use CityScripts if they were given tasks related to the Internet of Things and Smart City sensor integration, participants generally expressed that they would like to use CityScripts for these purposes. This was also mirrored in semi-structured interview sessions held at the end of the evaluation sessions.

5.3 Discussion

The questions that were to be addressed by the user evaluation are 1. Is the CityScripts system and the functionality it provides accepted and appreciated by users? 2. What are the benefits and points for improvement for CityScripts from a usability perspective?

Regarding the first question, indicative questionnaire responses and feedback from semi-structured interviews indicate that the CityScripts system was accepted and appreciated for the functionalities it provided to the participants. At the current stage of development, CityScripts is mainly appreciated for the functionalities it provides – this extended both to the basic functionalities and, for some participants, also to the advanced functionalities provided, which were described as making the system powerful and flexible. It is not perceived as very easy to use though, especially when interacting with advanced functionalities such as scripting or the Situation Awareness Framework, and also not as a “fun” system

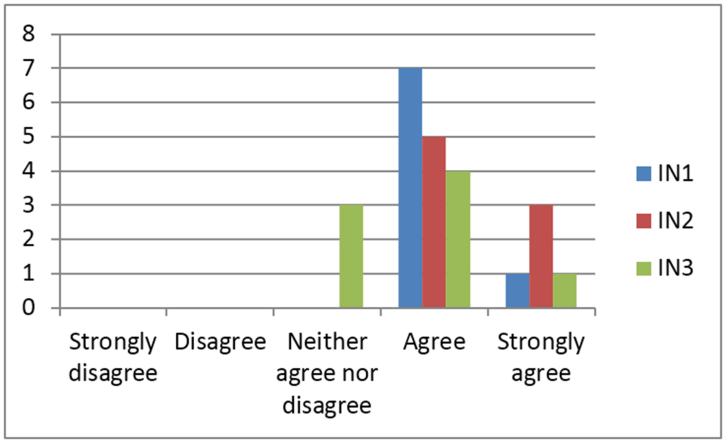


Figure 12: Intention to Use response distribution.

to use.

The response to the first question already contains a number of benefits that participants perceived when using the CityScripts system: both the provided basic and advanced functionalities of CityScripts were described as useful and strongly preferable to employing custom solutions for the purposes for which CityScripts was used during the evaluation sessions. The main points for improvement revolve around the usability of the system. Here in particular user guidance and instructions provided as part of the interface and the integration of scripting and the Situation Awareness Framework as advanced functionalities should be reviewed.

On a more general level, several participants suggested that the system might either be split into separate systems, where one provides a user-friendly interface but does not require users to add code themselves, and another could provide users with more programmatic access to the CityScripts functionalities – either within an improved user interface for advanced system functionalities or without a user interface and by using CityScripts APIs and user's preferred development environments instead. Several participants also suggested that an "intermediate" level of difficulty with more guidance for users and more akin to visual programming might be a suitable direction for future development of the CityScripts system.

6 Conclusion

This article introduces the CityScripts project goals, achievements and a formative user experience evaluation. Leveraging the deployment of an advanced IoT sensor infrastructure in the city of Santander, CityScripts provides end users with set of simple yet flexible tools that combine the Internet of Things, Services and People. Expanding on this, CityScripts furthermore provides powerful advanced features targeting advanced system users, including scripting support, Abstract Regions and situation-aware complex event processing, thus facilitating the implementation of powerful advanced functionalities. With this, CityScripts aims to empower end users and developers to access and productively use smart city infrastructures in order to create their own products and services.

Both the implementation and the formative user evaluation that has been carried out as part of the CityScripts project show the flexibility and intuitiveness of the main concepts underlying the CityScripts system. The implementation of advanced features including scripting and the inclusion of Complex Event Processing furthermore demonstrate how advanced functionalities can be realised on a technical level and can be integrated into the user interface of the system. Concerning the latter point, the evalu-

ation has shown that some work remains to be done in order to make CityScripts a compelling offer for early adopters.

More significantly, the evaluation raises the question whether a "one size fits all" user interface for interacting with smart city sensors and actuators should be preferred over providing separate modes or systems for basic and for advanced and more complex functionalities, for instance by exposing and documenting a CityScripts API instead of integrating features directly into the web-based CityScripts user interface. Further research and evaluations will be required in order to determine suitable solution approaches concerning these questions.

Overall, the CityScripts project provides a useful demonstration of how IoT and smart city components can be seamlessly integrated with online services. The infrastructure developed in the SmartSantander project was employed successfully and with limited effort required for the integration with CityScripts, which validates the technical implementation choices made both by the CityScripts team and by the SmartSantander project consortium as a whole, and feedback from evaluation participants indicates that the opportunities opened up by the combination of CityScripts and SmartSantander are appreciated and considered as useful. Future work may focus on making advanced features more accessible and on making the experience of using the system more fun, which may encourage quicker uptake from early adopters and more casual potential users alike.

Acknowledgments and Additional Notes

The CityScripts experiment has been co-funded by the European Commission as part of the SmartSantander project under contract number FP7-257992. The LinkSmart middleware has been co-funded by the European Commission as part of the Hydra Middleware EU FP6 project under the contract number IST-2005-034891.

Authors are listed in alphabetical order.

References

[1] J. Hernandez-Munoz, J. Vercher, L. Muñoz, J. Galache, M. Presser, L. A. H. Gomez, and J. Pettersson, "Smart Cities at the Forefront of the Future Internet," in *The Future Internet - Future Internet Assembly 2011: Achievements and Technological Promises, LNCS*, J. D. et al., Ed., vol. 6656. Springer-Verlag, 2011, pp. 447–462.

[2] D. Carboni, A. Pintus, A. Piras, A. Serra, A. Badii, and M. Tiemann, "Scripting a Smart City: the CityScripts Experiment in Santander," in *Proc. of the 27th International Conference on Advanced Information Networking and Applications Workshops (AINA'13 Workshops), Barcelona, Spain*. IEEE, March 2013, pp. 1265–1270.

[3] SmartSantander, "The smartsantander fp7 project web site," [online], 2013, http://www.smartsantander.eu, accessed April 2013.

[4] A. Pintus, D. Carboni, and A. Piras, "Paraimpu: a Platform for a Social Web of Things," in *Proc. of the 21st International Conference on World Wide Web (WWW'12 Companion), Lyon, France*. ACM, April 2012, pp. 401–404.

[5] LinkSmart Consortium, "Linksmart middleware," [online], 2013, http://sourceforge.net/projects/linksmart/, accessed April 2013.

[6] JBoss Inc, "JBoss Drools Business Logic Integration Platform," [online], 2013, http://www.jboss.org/drools/, accessed April 2013.

[7] A. Badii, M. Crouch, and C. Lallah, "A Context-Awareness Framework for Intelligent Networked Embedded Systems," in *Proc. of the 3rd International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies and Services (CENTRIC'10), Nice, France*, August 2010, pp. 105–110.

[8] Cosm, "Cosm," [online], 2013, http://www.cosm.com, accessed April 2013.

[9] IFTTT, "If This Then That," [online], 2013, http://www.ifttt.com, accessed April 2013.

[10] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, "From the Internet of Things to the Web of Things: Resource-Oriented Architecture and Best Practices," in *Architecting the Internet of Things*, D. Uckelmann, M. Harrison, and F. Michahelles, Eds., 2011, pp. 97–129.

[11] Yahoo!, "Yahoo! pipes," [online], 2013, http://pipes.yahoo.com, accessed April 2013.

[12] M. Welsh and G. Mainland, "Programming Sensor Networks Using Abstract Regions," in *Proc. of the 1st Conference on Networked System Design and Implementation (NSDI'04), Berkeley, California, USA*, vol. 1. USENIX Association, March 2004.

[13] A. Pintus, D. Carboni, and A. Piras, "The Anatomy of a Large Scale Social Web for Internet Enabled Objects," in *Proc. of the 2nd International Workshop on Web of Things (WoT'11), San Francisco, California, USA*. ACM, June 2011.

[14] A. Badii, "User-Intimate Requirements Hierarchy Resolution Framework (UI-REF)," in *Proc. of the 2nd European Conference on Ambient Intelligence (AmI'08), Nuremberg, Germany*, November 2008.

[15] A. Badii, D. Carboni, M. Tiemann, and A. Pintus, "CityScripts Deliverable D1.3 Final Report on Experimentation, Evaluation and Potential Improvements," SmartSantander Project, Reading, UK, Tech. Rep., 2013.

[16] J. Kim and S. Forsythe, "Adoption of Sensory Enabling Technology for Online Apparel Shopping," *European Journal of Marketing*, vol. 43, no. 9/10, pp. 1101–1120, 2009.

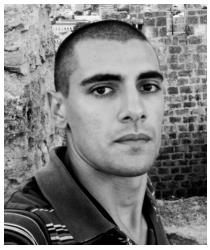
[17] H. V. der Hejden, "Pe-TAM: A Revision of the Technology Acceptance Model to Explain Website Revisits," *Serie Research Memoranda*, vol. 2000-29, September 2000.

Atta Badii, Senior Professor of Secure Pervasive Technologies, is the founding director of the Intelligent Systems Research Laboratory at the University of Reading and Hon. Distinguished Professor of Systems Engineering and Digital Innovation, University of Cordoba. Atta has multi-disciplinary academic and industrial research experience. His research stands at the confluence of intelligent interactive systems, cognitive robotics and human agent modelling as informed by the well-established principles of cognitive control engineering. He has over 25 years of experience in ICT, leading research both in industry (e.g. Schlumberger Technologies) and academia. He has over 200 publications to date and has served on various editorial and research steering boards as coordinator and expert e.g. as the Chair of the Security Architectures and Virtualisation Paradigms Taskforce of the European RoadMap Project SECURIST, the Director of The European Video-Analytics Network of Excellence for Socio-ethical and Privacy-respecting Video-Analytics (VideoSense), and, the Coordinator of the European Observatory for Crowd-Sourcing (ScieCafe2.0). Atta has contributed to over 25 large collaborative projects to date and has secured more than 190 Million Euro in funding through project and research grants.

Davide Carboni obtained the degrees MSc in Electronics (Università di Cagliari) and PhD in Informatics (Université de Sherbrooke). Since 1999, he has worked as a technologist in the ICT department at CRS4 . His main research interests include Internet-of-things, distributed systems, and indoor navigation techniques. Currently he is working as the head of research in Location and Sensor based computing Program at CRS4 and the coordinator at "Geoweb e Mobile Experience" Laboratory. He is the co-author of several papers in international referred conferences and peer-reviewed journals. He is a supporter of free/open source software and co-founder of the ApritiSoftware! association.

Antonio Pintus obtained the MSc degree in Computer Science Technologies. Since 2000, he works as a technologist for the ICT Department at CRS4. His skills include the design and development of distributed Web-based software systems, Service-oriented Architecture (SOA), and REST-based systems. Currently he is working in the field of Internet of Things and Web of Things. He has several publications and scientific papers presented in national and international conferences. He serves as a software architect and backend developer in Paraimpu.

Andrea Piras holds the Master Degree in Information Technology (M.Sc.) at the University of Cagliari and from 2000 he worked in the ICT group of the CRS4 contributing to national and international research projects. His main activities focus on Human-Computer Interaction (HCI), Graphical and Tangible User Interfaces (GUI - TUI), and Augmentative Alternative Communication (AAC), integrating components of multi-platform, Web and context-aware. He has published several publications and scientific articles in national and international conferences while serving as a reviewer for international journals and conferences.



Alberto Serra holds the M.Sc. in Computer Science Technologies. He authored publications and scientific papers. Since 2007, he has worked as a consultant in the ICT group at CRS4, taking part in projects related to mobile computing, geo-referenced applications, indoor mobile personal navigation systems and Internet of Things. Currently he is focusing on sensor data based projects for the Web of Things and mobile applications.

Marco Tiemann completed his studies in Information Science at the University of Hildesheim, Germany, in 2003. Before joining the Intelligent Systems Research Laboratory at the University of Reading, United Kingdom, in 2010, he worked at the Philips Research Laboratories in Eindhoven, The Netherlands, as a Research Scientist. He is a member of the ACM and the IEEE. His main research interests are machine learning methods applied in challenging real-world usage scenarios.

Nagarajan Viswanathan In 2003, Nagarajan completed his BEng Degree in Computer Science & Engineering with 1st Class Honours and Distinction from Manonmaniam Sundaranar University, India. He joined the University of Edinburgh, UK to pursue his research interests through postgraduate study and concluded his MSc Degree as one of the highly-ranked students of the course. Nagarajan has built a track record of industrial experience in software engineering and project management; this includes for example work with Cognizant Technology Solutions and Infosys Technology Solutions. His research interests include Robotics, machine learning, multi-agent problem solving and knowledge representation.