iDASH

An enhanced middleware for an ad-hoc service-oriented Smart Home

Bruno José Furtado Gonçalves

Instituto Superior Técnico, Technical University of Lisbon, Portugal

brunojfgoncalves@ist.utl.pt

Abstract

The recent and sudden proliferation of smart devices in home environments has led to poor interoperability between them. Furthermore, the absence of intuitive interfaces and configuration in currently available systems requires users to have highly specialized knowledge to build and manage a smart house. For that reason, a platform of service oriented middleware was developed, allowing access to the functionality offered by any device or application in a residential environment, transparently, intuitively and regardless of the communication technologies involved.

Nevertheless, that platform still presented some limitations, mainly at the user interface and contexts management levels, making its use in real environments more difficult than acceptable.

The objective of this work was to continue development of the previously created solution, providing the existing platform with new features and making it more accessible to the everyday user.

This work follows three lines of development: it focuses essentially on the application layer created on top of the previously developed infrastructure and branches into, on the one hand, the development of new modules to be inserted into the platform and, on the other hand, the implementation of new features for the core system.

The system was deployed in a real test bed, allowing for the development of new services and interfaces improving user interaction. Performance and usability tests were conducted, with globally positive results.

Keywords: *Smart Home, Home Networking, Services, Interfaces, Contect, Middleware*

1. Introduction

Technology has always evolved towards improving mankind's day-to-day life and, as a result, we are now living in a society where human beings use, on a daily basis, a growing number of increasingly sophisticated devices and equipment, both in their professional and residential environments. The huge diversity of desired functionalities and the resulting increase in system complexity has motivated the search for solutions that enable device intercommunication, as well as fast and easy management of the controlled equipment.

In the meantime, the concept of Smart Homes arose, aiming to define houses which possess enough intelligence to autonomously adapt to specific needs of its inhabitants, providing a wide range of features, such as the monitoring/controlling of all its spaces. In order to make it possible, there is an internal network which connects every single device inside the Smart Home.

Lately, industry has been showing great interest in this area. As such, there has been a never ending stream of consortiums created to explore and standardize these type of networks, as is the case of UPnP (Universal Plug and Play Forum) and DLNA (Digital Living Network Alliance). However, there is still the need for better solutions in terms of functionality, manageability and interoperability between devices. In terms of functionality, the major need is in creating more efficient solutions regarding the identification and management of contexts, which allow maximizing adaptation of the system to the lifestyle of each user. Studies such as [***1***] and [***2***] also demonstrate that the acceptance of these systems by the general public strongly depends on the level of usability of the user interfaces, i.e. it is expected that the developed products provide an interface as user friendly as possible.

Recently a service-oriented middleware solution called DASH [***3***] was developed at Instituto Superior Técnico, in the scope of an MSc dissertation. This solution intends to handle the interconnection of devices, making it easier to install and integrate these same devices in a distributed network architecture. This is interesting from the

implementation point of view, as it makes the whole network more adjustable to the users' needs. Furthermore, since its development was based on a modular programming paradigm, it enables the deployment of new functionalities independently from the existing ones, thus assuring the interoperability between devices and applications. Nevertheless, for the solution to be usable in real environments, there is still some work to be done, requiring the development of new modules:

- A graphical interface that enables easy setup and management of network devices;
- Security mechanisms that guarantee information privacy and authentication of users and devices;
- A context identification and management subsystem;
- A user profile management subsystem.

The possibility of working on the evolution of the DASH system, improving its feature set, represents the motivation for the work being presented. The solution developed and presently described continues the work previously mentioned, but focuses on the application layer (which works on top of the existing infrastructure) and aims at the development of new middleware modules and of new applications, adding functionality to the system. It also encompasses the planning and deployment of a real testing environment, which in the future may be used to validate applications and to study the interaction of users with the system.

3. Architectural Overview

Starting with Figure 1 and taking into account the goals of this work, the logical architecture designed for the solution is presented on Figure 2. This new architecture includes the new modules inside the middleware (Context Manager, User Profile Manager, Device Profile Manager and Network Interface Manager) and outside the middleware (Graphical Interface).

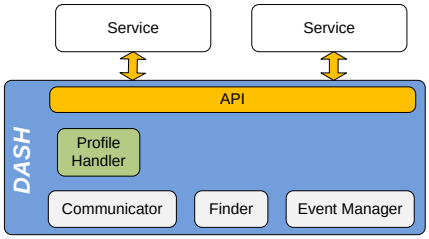


Figure 1 – Old DASH architecture

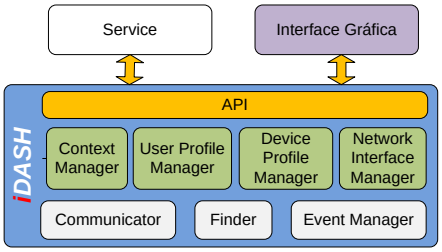


Figure 2- New logical architecture solution

3.1 Context Manager

This module collects the data generated from detected events, manages its processing – taking into consideration the parameters preciously defined – and, in response, triggers the appropriate actions, as illustrated in Figure 3.

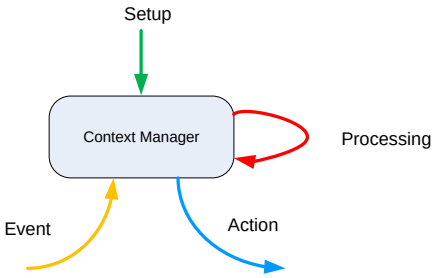


Figure 3 – Context Manager Module

Regarding Figure 3 and the fact that this solution is based on a distributed architecture, the following requirements are evident:

- This module needs to exist in various network nodes, to ensure the robustness of the system.
- The setup information needs to be distributed among different network nodes.
- Information about the network has to kept up-to-date, to ensure the reliability of the action to be taken in reaction to events.

3.2 User Profile Manager

The User Profile Manager module is responsible for adjusting the parameters, i.e., the means in which the system acts, according to the presence of each user, individually or together with others. This module provides information important for the Context Manager module to process events, as illustrated in Figure 4.

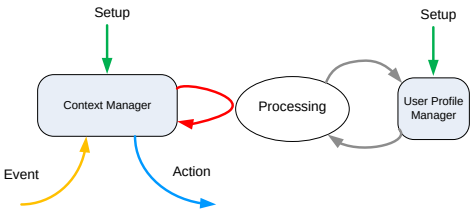


Figure 4 – User Profiling module

This module has the following set of requirements:

- As there is no need for all devices to have the profile information or users, and due to the limitation of some of the devices, this module cannot be presented on all of them.
- To ensure the consistency of information, it is necessary to disseminate all updates that are made in the profiles of users.
- As this module interacts with the Context Manager module, it becomes necessary to ensure that there is a well-defined interface to enable communication.

3.3. Device Profiling

This module, apart from managing all of the active services found in a device, also characterizes this same device in terms of its resources and the architecture, the operating system and the available network interfaces.

3.4. Graphical Interfaces

This module allows you to configure the parameters that are used by the system, view the status of devices and services and facilitate the system's utilization.

This module has the following set of requirements:

- Be accessible by a larger numbers of devices integrating the system (examples: Smartphone, computer, set-top box, etc.).
- Be accessible via Internet or SMS
- Remotely control some of the devices
- List all the services available across the network
- Allow setup, activation and deactivation of services available in each device

To facilitate the creation of new interfaces, a new module was designed – Interface Service – which provides a set of functions that can be used for future interfaces, as presented in Figure 5.

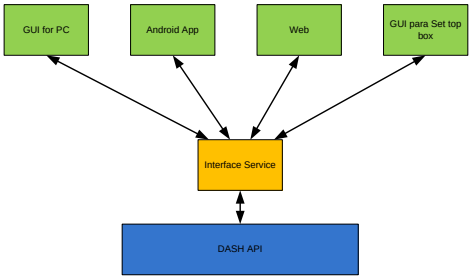


Figure 5 – Interface Service module

Regarding the requirements, these interfaces must be easy to use, as they should be intuitive for the common user, and functional, since they must run the commands properly.

3.5 Network Interface Manager

The Network Interface Manager module is responsible for selecting the used network technologies for each communication instance. Since some devices may support more than one technology, this module will identify the available ones and manage the physical connections to ensure the utilization of the most appropriate one by each service.

In Figure 6, the architectural model of a baseline system is presented with several nodes, all interconnected and running the software. Some nodes may support many network technologies, both for communicating with devices which connect to these nodes, and for communicating

with other nodes. The number and type of supported technologies may vary from node to node.

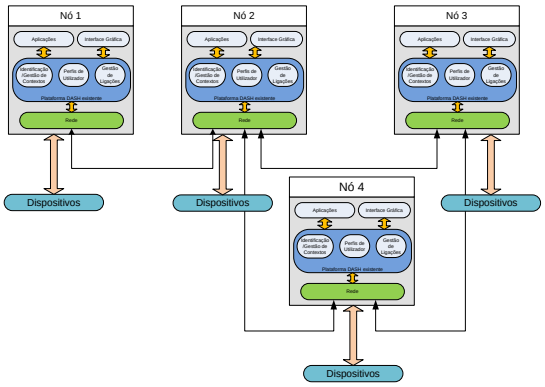


Figure 6 – Architectural model

When characterizing the several network connections that each device may hold, this module must choose the most adequate technology for a certain service. This module is also responsible for performing the *handover* between communication technologies when required.

Figure 7 demonstrates how the module operates. Basically, this module is notified whenever an action takes place and, in response, will indicate which network interface should be used. This module needs to hold updated information on the state of devices' available network connections, as well as information regarding the parameters that were defined.

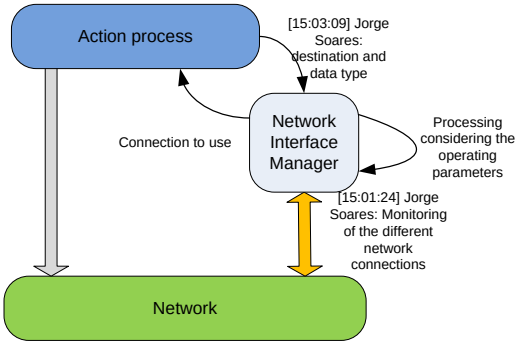


Figure **7** – Network Interfaces Manager Module

4. Implementation

In this section, proposed solutions for the implementation of each module will be presented, considering the requirements that were previously specified. In addition to the proposed final solutions for each modules, other alternatives are also discussed.

4.1. User Profile & Device Profile

These two modules have in common the fact that both hold system information, which has to be updated and kept consistent across all nodes in the system. The problem of accessing information in a distributed system can be resolved by various approaches.

A distributed file system could be created - as it was initially thought for the DASH system using Hydra File System [***4***] - and used to store all information. However, this approach introduces all the complexity inherent to these types of systems, which is not viable for some resource-limited devices, such as mobile phones and set-top boxes. Moreover, for the information to be kept coherent on the system, this type of solution requires that a minimum number of nodes are always present.

Another approach would be a centralized repository, which would reduce the level of complexity of whole system, since it would make the whole update process simpler. However, this solution goes against the main objective of this system, i.e., being distributed.

Due to the amount of information that is used in this system, a simpler solution - when compared to the previous ones - was designed. This solution consists of the creation of two files on each node (one containing information about users and another containing information about the device); all of these files are updated when events occur in the system, thus maintaining the consistency of information, and can be accessed by all others network nodes, when necessary. With this solution, the information is kept distributed and accessible to any device connected to the network.

In order to enable the use of these same files by other systems, the standard XML [***5***] (eXtensible Markup Laguage) was chosen. This standard was chosen as it is fairly easy to read and debug, and tools are available for most languages and platforms.

Herewith, the system now has two XML files (XMLUser and XMLDevice) in each device, as illustrated in Figure 8.

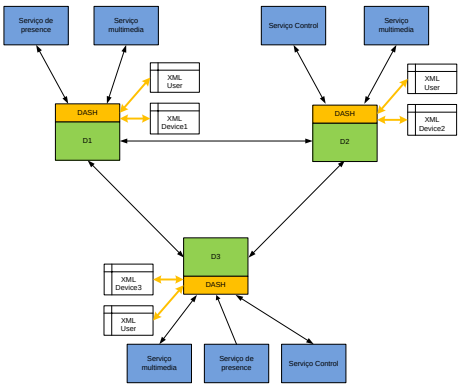


Figure 8 – Architecture with profiling files

To manage both profile files, two modules were created separately, UserProfiling and DeviceProfiling.

The UserProfiling module is responsible for managing all user information and ensuring that it is consistent. The DeviceProfiling module is responsible for managing the device information and characterizing the node, as is illustrated on following figure.

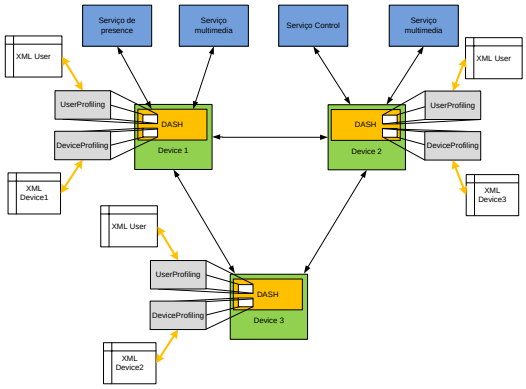


Figure 9 – UserProfiling and DeviceProfiling modules

4.2. Context Manager

This module, as mentioned in the architectural specification, is highly dependent on the modules UserProfiling and DeviceProfiling, to make the right decisions, by taking into consideration each user (RFID tag, luminosity level, playlist, etc.) and the current state of the system (active/inactive services, etc.).

Before this module was implemented, the system working according to Figure 10, where an event, created by a service, is receive directly by another service, which processed the event.

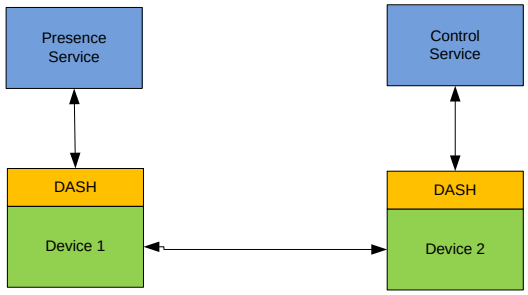


Figure 10 – Old operating mode

This mode of working is good for simple systems, but when a lot of services (some very similar between them) and devices are present, managing them all becomes very complex.

The solution adopted was the creation of a new module inside the middleware layer. When this module detects a new event, it processes it internally and then sends commands to others services, as shown on Figure 11.

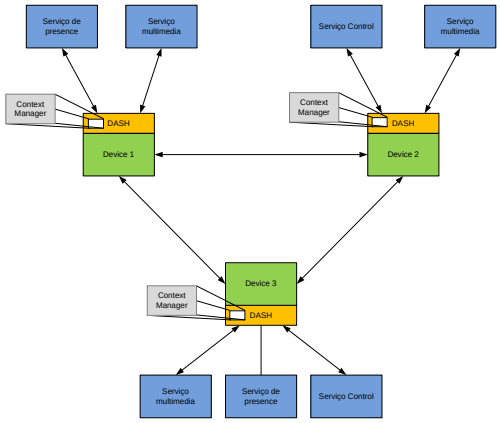


Figure 11 – Architecture with Context Manager

With this new system philosophy, the number of messages transmitted as a result of the occurrence of a given event has grown significantly. However, the negative impact of this growth is not significant when the size of the messages (typically less than 200B), their low frequency and the available network bandwidth available (typically of the order of Mbps) are taken into consideration. In this architecture, the services become mere recipients of commands and/or event generators.

4.3. Interfaces

To simplify the creation of new interfaces, a new module was built - InterfaceService –, as mentioned in the architecture overview.

Three interface prototypes were developed, two graphical (a GUI for computers and a Web Interface that can be accessed by any device providing a web browser) and one based on the Wii Mote (where it is possible use its 3D accelerometer). As an example, the Web Interface built is presented in Figure 12.

Figure 12 – Web Interface

5. Evaluation

To show that the middleware which has been developed can support multiple applications, which in turn can be developed over the middleware layer, a set of services and an experimental test-bed were created.

Figure 13 shows the real environment, which was created and assembled at IST.

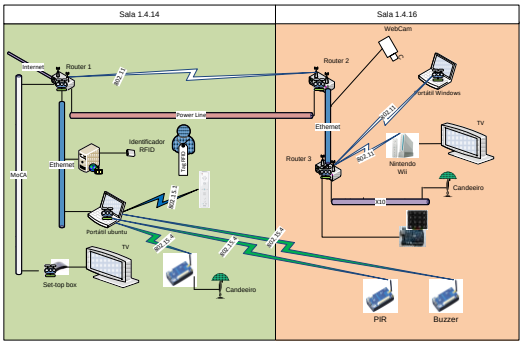


Figure 13 - Test-Bed

As evident in Figure 13, the test-bed is composed of a vast number of devices and communication technologies. Several types of services were developed, including multimedia, presence, control and advertising functions.

5.1. Performance tests

To get an idea of the performance of the Context Identification and Management module on the different equipments in the testing environment, a generator of random presence events has been developed, , allowing stress testing of the module. The event generator sent five thousand events to the system, at a medium average speed of 140 events per second. At this point, the necessary time to process each event, both on the RouterBoard 433UAH and the computers, under Ubuntu and Windows XP operating systems, was measured. From these measurements, the graphics illustrated in Figure 14 and Figure 15 was built based on the obtained data.

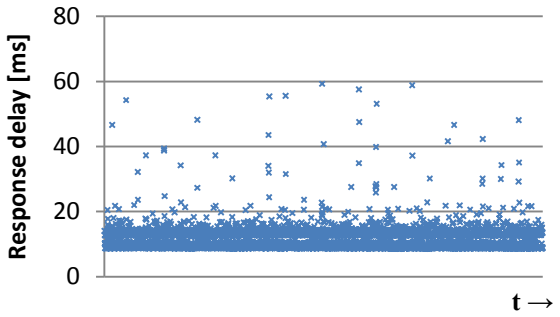


Figure 14 - Computer with Ubuntu

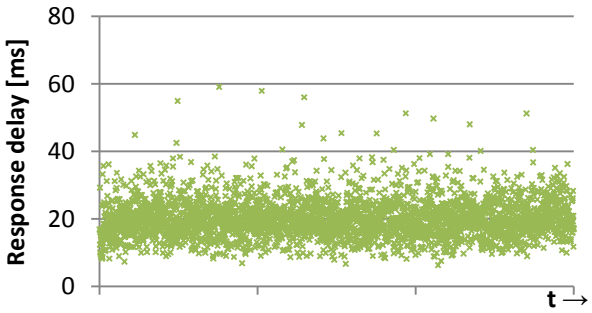


Figure 15 - Computer with Windows XP

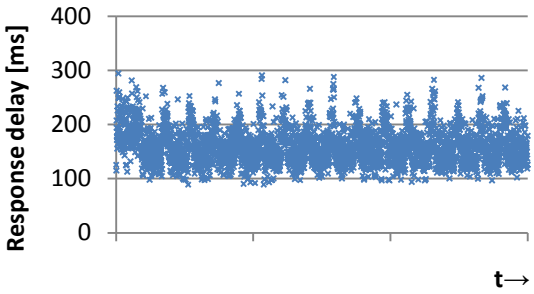


Figure 16 - RB433 with OpenWRT

After analyzing the obtained graphics and the values presented in following table, it is noticeable that the delay is significantly higher on the RB433. This was expected, since it is an embedded system with fewer resources than an actual computer.

	Computer (Ubuntu)	Computer (Windows XP)	RB433 (OpenWRT)
Average [ms]	12,56	19,37	161,54
Deviation pattern [ms]	4,38	6,67	34,19

However, this data is not entirely conclusive as to the module's performance. Hence, another test was performed, to try to understand what is the maximum message rate at which the system can work. This consisted of sending one thousand messages, at different rates, and measuring the time elapsed between the generation of the event and the end of the processing of this same event by the Context Manager module. The graphic illustrated in Figure 17 was obtained.

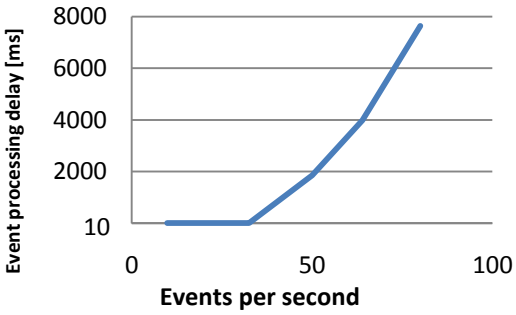


Figure 17 – Context Manager Module performance analysis

Analyzing this graphic, we may find out that from 30 events per second up, the system response becomes slower. Taking into account the domestic environment for which the system is destined, this limit is perfectly acceptable, as situations where events occur with such high rates are rare, and some delay in those situations is tolerable.

5.2. Usability tests

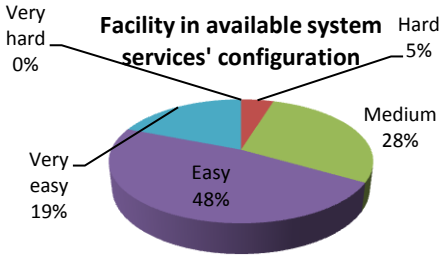
In order to assess user interest in the developed system, as well as the set of services and developed interfaces, a survey was carried out on a small group consisting of twenty one students of Instituto Superior Técnico who interacted with the testing environment.

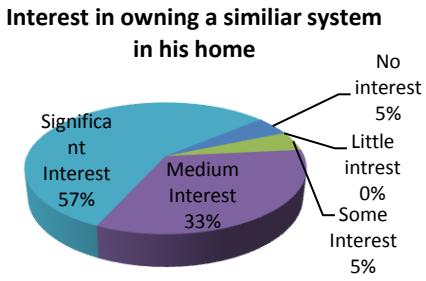
During the interaction, users had to configure the different services in the system, using the interfaces available in the various devices (Wii Mote and Web Interface via Nexus One, Wii and computer).

In the inquiry, the following set of metrics was considered:

- Ease of understanding the system functioning;
- Ease of configuration of available system services;
- Perception of the system's response time;
- Wii Mote Interface;
- Web Interface.

Some of the results obtained from the survey are illustrated below:





It is possible to reach the conclusion, looking at these results, that this system is easily understandable by ordinary users and presents a good response time. On the other hand, users were less pleased with the services' configuration, as they seemed to find this point more confusing.

As for the interfaces, in spite of being early and simple versions, they proved to be visually appealing and functional to its users, most of which ranked them as "good" or "very good".

Lastly, it is clear that a great percentage of the participating users seemed very interested in purchasing such system.

6. Conclusions and Future Work

The *Smart Homes* area has seen great development in the last years, both in the commercial and R&D spheres. That led to the implementation of a large number of solutions that are still lacking in interoperability, coverage and user-friendliness.

The main purpose of this work was to continue the DASH project, which was previously developed at Instituto Superior Técnico. It proposed a *middleware* based solution that allows the intercommunication and management of devices of a network in intelligent implementations as the *Smart Homes*.

This work's purpose was to make the developed platform more complete and improve its user-friendliness. In order to do so, the proposed solution implements new functionality, user interfaces, connection management features and means for identification and management of contexts, taking into account the user profiles and the services made available by the different devices.

The following modules were implemented:

- Context manager (for identification/management of contexts)
- Device Profiling (for the management of resources and services available on each device)
- User Profiling (for the management of the users profiles)
- Interface Service (supplies a set of function that can be used to create new system interfaces)

The following interface modules were also implemented: computer GUI, web interface and Wii Mote-based.

To assess the system, a test environment was created, with a vast set of devices where it was possible to implement services, allowing for functional level validation. Some performance tests were also conducted, which showed that the system has a pretty satisfactory behaviour, taking into account the environment it was design to work in.

At last, the system usability was tested by a group of users, which interacted with it and later answered a small survey. The results show that both the implemented system and its interfaces were considered very interesting to the majority of the users.

Analyzing the results obtained at the end of this work, all the proposed objectives seem to have been met. The DASH platform became user-friendlier and validation tests took place.

When assessing the work, some goals for possible future work were identified:

- Integration of the platform with other systems;
- Integration of new technologies, such as DLNA;
- Development of applications for Android and iPhone platforms, very popular interfaces.

7. References

[1] W. Keith Edwards and Rebecca E. Grinter, "At Home with Ubiquitous Computing: Seven Challenges," *Ubucomp 2001: Ubiquitous Computing*, pp. 256-272, 2001.

[2] Scoot Davidoff, Min Kyung Lee, Charles Yiu,

John Zimmerman, and Anind K. Dey, "Principles of Smart Home Control,".

[3] Pedro Rebelo, "DASH: A distributed service-oriented middleware for ad-hoc home netwroking environments," Instituto Superior Técnico, Dissertação de Mestrado 2009.

[4] B González and G K Thiruvathukal, "The Hydra Filesystem: A Distributed Storage Framework," , 2006.

[5] W3C. [Online]. http://www.w3.org/