The STORM Use Case: Leveraging IoE for the Preservation of Cultural Heritage

Sophia Robinson[1], Mason Robinson[1], Elijah Brown[2]
[1]University of West Attica, Greece [2]Engineering Ingegneria Informatica, Italy

ABSTRACT

Climatic changes and intensive industrialization, have contributed to increasing the risk of damage of Cultural Heritage (CH) artefacts. On the other hand, small and medium sized museums, as well as small CH sites struggle to fulfil international recommendations for protection and conservation, due to budget limitations. The constantly increasing potential of IoT enabled devices and the establishment of cloud technologies as an enabling framework can help address this issue. In this paper, we present an Internet of Everything (IoE) architecture, empowered by an easy to deploy cloud framework for the protection of CH. Particular use cases from CH sites are presented, as these have been identified in H2020 STORM project for safeguarding Cultural Heritage through technical and organizational resources management.

Keywords: Internet of Everything, Cultural Heritage, Cloud Computing, Edge Computing, Internet of Things, Data Flow, Platform Architecture Design, Requirement Definition

INTRODUCTION

Preservation and protection of the Cultural Heritage (CH) have always attracted the attention and interest from many parties (e.g., academics, researchers, local or domestic communities or private companies and the public sector) due to its historical, touristic and economic potentials. The recent advances in technologies, in particular the rapid growth of Cloud Computing and the Internet of Things (IoT), have been the driving force behind a great development on the techniques and the proposed solutions that are considered as efficient to be used for the preservation and protection of the Cultural Heritage. The use of (many kinds of) sensors that are able to communicate intermittently and store the data to a database that can be located locally (i.e., edge of the network) or in the cloud, and where they can be accessed by any expert, registered individual has been considered as a, potential, basic scenario in a modern Cultural Heritage use case.

Figure 1 presents the entities of Internet of Everything (IoE) that have as their goal the safeguarding of Cultural Heritage. In particular, the IoE entities are the following:

- **Things:** IoT devices (e.g., sensors) used for weather (e.g., air temperature), environmental (e.g., audio signals) and structural monitoring (e.g., material degradation) of the parameters linked to the risks that the cultural heritage site faces.
- **People:** Human-beings (e.g., experts) connected to each other through the use of social networks or crowdsourcing applications.
- **Data:** Incoming data (from IoT devices or people) are processed to extract useful information and detect hazardous events, in order to make decisions.
- **Process:** The stakeholders are informed on real-time about hazardous events through an emergency management system.

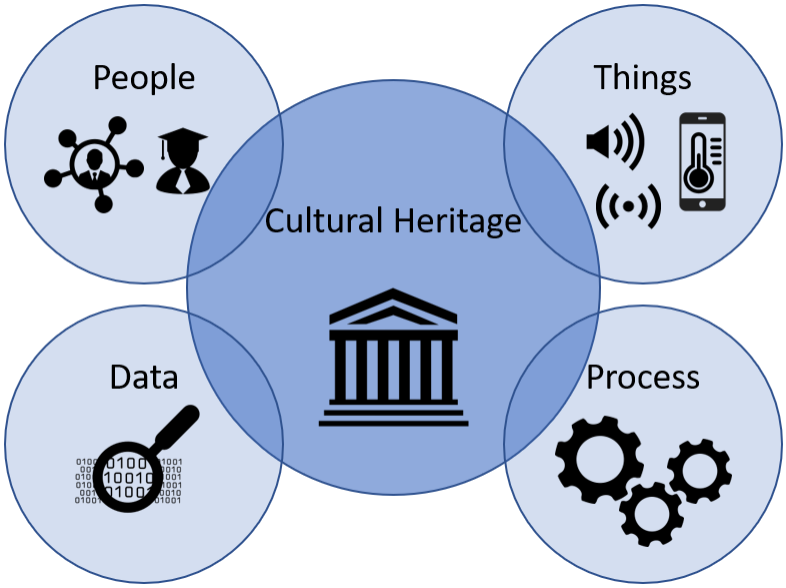

Figure 1. Internet of Everything for Safeguarding Cultural Heritage

To this end, in this chapter, we will present the logical architecture of an Internet of Everything (IoE) system that can be used to increase the preservation and protection of the Cultural Heritage, as this is the case in STORM, an HORIZON 2020 funded European Project on Cultural Heritage. Before describing the requirement specification process and the design of the architecture, a section that briefly describes how the recent technological evolution has affected CH along with examples of other studied architecture proposals and their performance.

BACKGROUND

Given the importance and the social impact that the protection, preservation and conservation of humanity's Cultural Heritage has to the modern communities, the recent advances in Information and Communication Technologies (ICT) have led to a study about the technologies that can play a significant role in this effort. Especially, the rise of the monitoring and communication capabilities of the modern low-power, high usage sensors, and their ability to connect (often through the Internet) and operate under any physical conditions and location by creating advanced ecosystems of smart things has been considered as a potential solution with many applications in the Cultural Heritage field. In this section, a description of various proposed solutions where IoT (or IoE) architecture and/or systems have been considered for the preservation or protection of Cultural Heritage will take place.

An idea for the architecture of a platform that is powered by IoT capabilities is presented in Chianese et al. (2016) describing, also, a possible use case that demonstrates how IoT characteristics can enhance the experience of visitors in an archaeological site. The authors described how, based on their proposed architecture, they have developed an application for mobile phones (i.e., named TolkArt) that will allow the visitors to receive multimedia content on their phone containing images of near-by exhibits, description of the exhibition of their selection and multimedia files and artwork that simulate the one-way interaction with the artefact. Finally, the visitors can use the application to describe their feelings about the visit and the enjoyable presentation that is provided to them in a forum-like page of the application.

In Chianese et al. (2015) the authors build on the previous idea and present an updated version of the application along with more details on the sensor network that is needed for the expected system performance. In addition, deeper analysis of the visitors' behavior is also provided, where the results manage to show that the overall experience is better rated and that the overall time that is spent on the tour has been significantly increased, showing the raise of interest that the use of this IoT solution managed to create. A similar approach is, also, described in Chianese et al. (2013), where the authors propose a location-based mechanism integrated with the mobile application could be used in order to provide the details, in the form of images and/or text, of the nearest exhibits.

The authors in Ott and Pozzi (2011) study how does the role of the ICT, and the recent improvements on the field, manage to change and affect the teaching and learning of the Cultural Heritage. Digitization of the artefacts is regarded as a very important step towards strongly involving ICT solutions for improving Cultural Heritage, since it is considered as an initial step to provide personalize experience to the visitors through mobile applications and gamification techniques. Finally, the authors propose that ICT could improve the interdisciplinary learning approaches since it will facilitate the access to data from different sources that could be combined in a way to provide general remarks that could, also, be used in different sites under similar weather or natural conditions.

Another reference IoT architecture, aiming at the preventing conservation of Cultural Heritage artefacts focusing on the use of sensing for this purpose is presented in Perles (2018). More specifically, a solution that includes the deployment of LORA sensors is proposed that will communicate with Sigfox technologies (Sigfox, 2018). The authors study the performance of this solution aiming at receiving optimal results regarding the avoidance of detrimental effects on the artwork of the exhibits in a Cultural Heritage site. The combination of LORA with Sigfox was inspired by their use in many tourism applications at the smart cities domain, an area that has also been highly benefited from IoT applications.

Finally, in Minerva (2014), the author introduces the meaning of Servitization as the result of virtualization in an IoT environment. Servitization allows the transformation of real-life objects and products into services that can be delivered by a programmable platform. Especially, the idea of the Virtual Continuum that is the digital representation of almost any real-life object has attracted the attention of the researchers on the Cultural Heritage domain that study solutions where this notion is placed in motion, basically by proposing methods for visitor- (digital) artefact interaction.

DESIGNING AN IOE PLATFORM ARCHITECTURE FOR CULTURAL HERITAGE

This section presents the proposed IoE architecture starting from specification of requirements that have led to the decision in the design. For better understanding, those requirements have been divided in two different categories: Functional requirements that are closer related with the system requirements (depending on the project CH site's needs) and the Non-Functional requirements that are closer related with the user needs.

Definition of Functional and Non-Functional requirements

The first step taken towards the design of the STORM platform's architecture, was the identification of the use case requirements. In particular, we identified the use case scenarios, based on a detailed analysis regulatory framework, standards and classification work for Cultural Heritage sites, and the particular

profiles, needs and expectancies of participant sites.

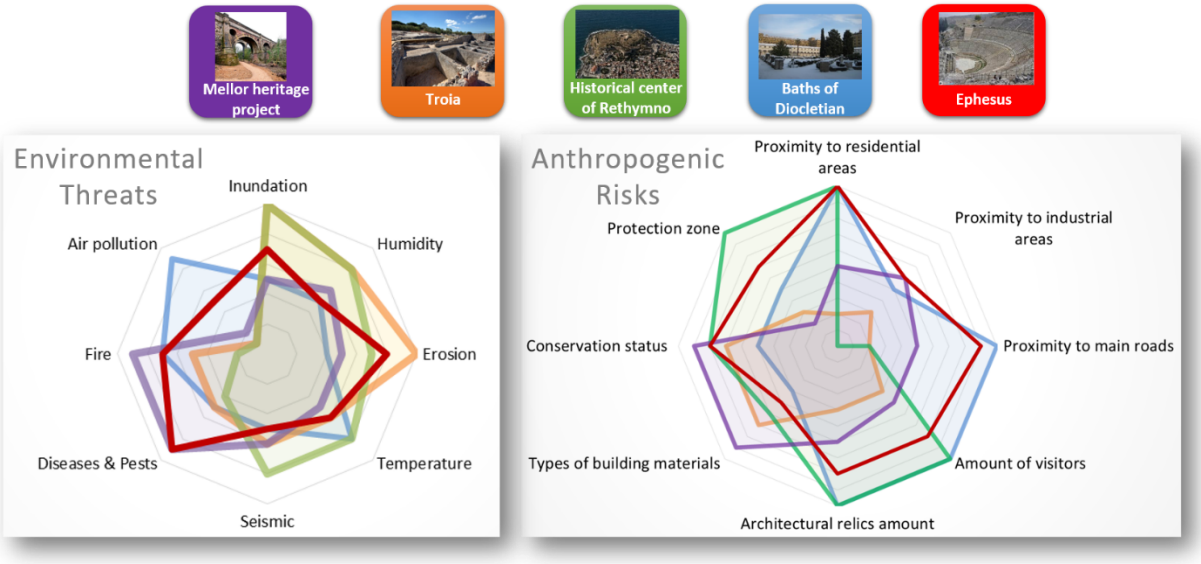

Figure **2** illustrates the importance of all anthropogenic risks and environmental threats for each participating site in the STORM project.

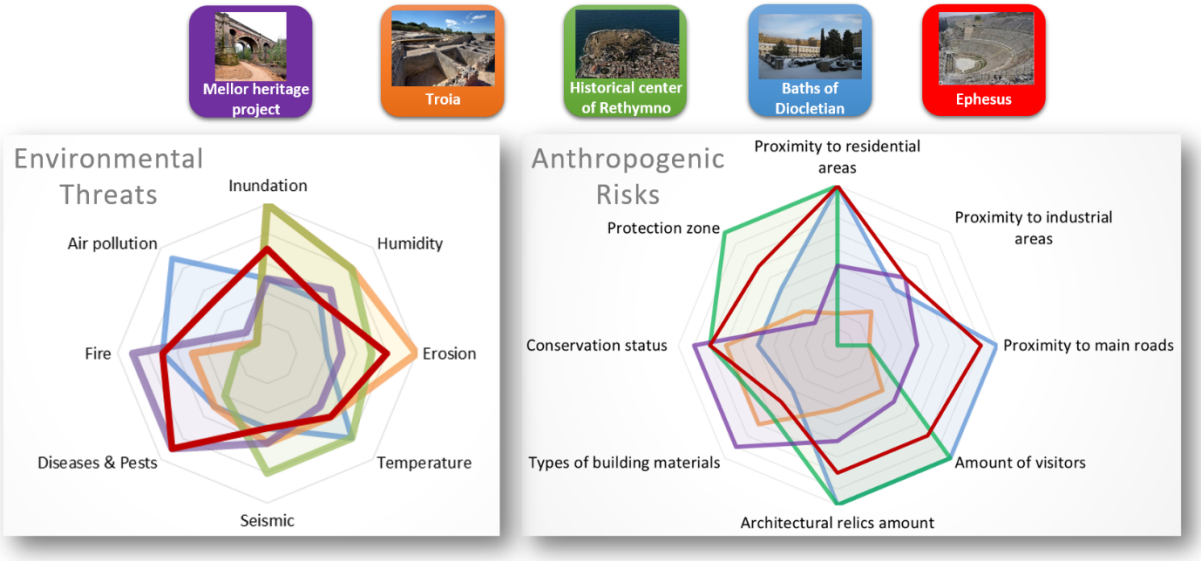

Figure 2. Pilot sites profiles according to a) environmental threats, b) anthropogenic risks.

Each pilot site (i.e., Mellor Heritage Project, Historical Centre of Rethymno, Baths of Diocletian, Roman Ruins of Troia and Ephesus) produced from two to five use cases for covering their needs and hazards in terms of both natural hazard protection and climate change effects mitigation. Table 1 presents all the parameters we selected to define a use case.

Table 1. Description of the parameter that define a use case

Parameter	Definition

Name		Specifies the name of the scenario, which provides an abstract idea on what it addresses.
Identifier		A unique identifier for each scenario, consisting of letters referring to the site and a unique number per scenario per site.
Description	What	Describes the artefact or structure to be protected, restored or intervened.
	Where	Describes the location and particular conditions where the artefact, structure is located or stored.
	Why	Describes the particular reasons/risks, which drove us to select this use case.
Scope		Defines all the means that will be used to achieve the goal.
Goal		The goal to be achieved.
Requirement		These are the preliminary user requirements for the use cases are listed here.
Assumptions		Specifies what is needed in order to achieve the use-case goal in terms of technical assumption (e.g., existing sensors and infrastructure) and organizational assumption (e.g., existing techniques, processes, policies, regulations, etc.).
Post conditions		The final conditions and linked results expected.
Technical Dependencies		Presents the technological solutions, already existing and to be developed, that will be needed for this use case.
Human Actors		Presents the human actors that are involved in this use case (i.e., firefighters).
Exceptions		Defines any exceptions that may influence the good execution of scenarios related to the use case.

During the definition of use cases, the preliminary (non-technical) user requirements were recorded; they are used to guide the formulation of functional and non-functional requirements. The parameters that define a preliminary user requirement are the following:

- **Req_ID**, which is a unique identifier per requirement using the schema *REQ-<pilot site>-<process action>-<sequential number>*. The field *pilot site* is referred by three letters (e.g., BOD for Baths of Diocletian), the field *process action* is referred by another three letters (MON for monitoring, MAN for management, INT for intervention) and the *sequential number* is the index of the requirement.
- **Relevance**, which is defined by using MUST (for mandatory requirements), SHOULD (for desirable requirements) and COULD (for optional requirements).
- **Owner**, who is involved in the requirement definition (matching with project objectives and needs) and implementation (matching with system functionalities and components).
- **Description**, where a short explanation of the requirement is provided.

For example, the use case *Halls I, II and IV* placed in Baths of Diocletian produced the requirement that the environmental information must be collected by capturing sounds (Table 2).

Table 2. Preliminary requirement for capturing environmental sounds

Req_ID	Relevance2	Owner	Description
REQ-BOD-MON-02	M	TEIP	Information about the environmental conditions MUST be collected by capturing characteristic sounds accompanying corresponding events (e.g., local storm pattern and lightning thunder).

As a result, the aforementioned requirement led, for example, to the formation of the functional requirement that classification techniques should be adopted to define the source of the sensor signals

(e.g., thunderstorm) and the non-functional requirement of sensor reliability (Table 3**Table 3. Example of functional and non-functional requirements**).

Table 3. Example of functional and non-functional requirements

Req_ID	Description	Preliminary Req_ID
FR-IE-02	Classification techniques MUST be performed on **sensor** signals to define its source.	REQ-BOD-MON-03
NFR-Rel-02	Reliable sensors SHOULD be used to measure the environmental parameters.	REQ-BOD-MON-03

In Table 3, the column Req_ID provides a unique reference identifier per requirement using the following format: *<type of requirement>-<requirement's acronym>-<sequential number>*. The type of requirement is indicated by FR for functional requirements and NFR for non-functional requirements. The requirement's acronym is indicated by two or three letters, usually the first letter of the requirement's name. For example, for the information extraction requirement we selected IE. Finally, the sequential number denotes the index. It should be noted, that the same requirement may apply to more than one use case.

For giving a short description of the requirements we used the MUST, SHOULD, COULD verbs to define the degree of necessity (IEEE 1998). MUST verb (Essential) is used to indicate the platform will not be acceptable unless these requirements are provided in an agreed manner; SHOULD verb (Conditional) is used to imply that these requirements could enhance the software product, but it is not unacceptable if they are absent; COULD verb (Optional) is used to indicate a desirable requirement.

Finally, in order to respond to the collected system and user requirements defined we specified the system architecture for the STORM platform. For example, the *FR-IE-02 requirement* is addressed by the raw Data Pre-processing, Feature Extraction and Data Processing modules.

The whole process for designing the system architecture (Figure 3), follows the recommendations of IEEE (IEEE 1998), by ensuring the traceability of the specified requirements, covering both backward and forward traceability, as follows:

- Backward traceability (i.e., to preliminary user requirements), since each requirement refers its source in earlier documents.
- Forward traceability (i.e., to platform architecture), since all the defined requirements have a unique name and can be referred without ambiguity in the following documents.

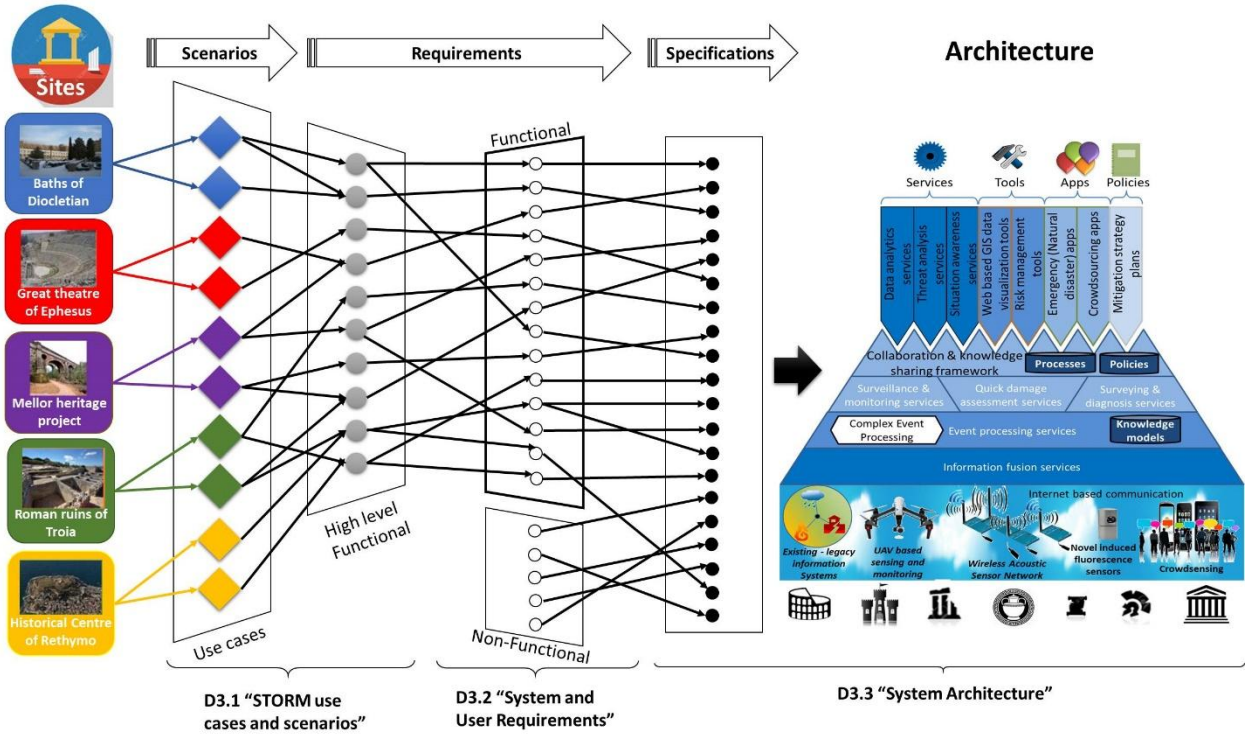

Figure 3. Design process of STORM platform architecture

Platform architecture

Having discussed the Functional and Non-Functional requirements, an architecture is designed that manages to efficiently address those requirements. In addition, the proposed architecture in STORM is designed with respect to the IoE entities (i.e., People, Data, Process, and Things) and consists of 6 logical layers: the Application, the Service, the Event, the Information, the Data and the Source Logical layer.

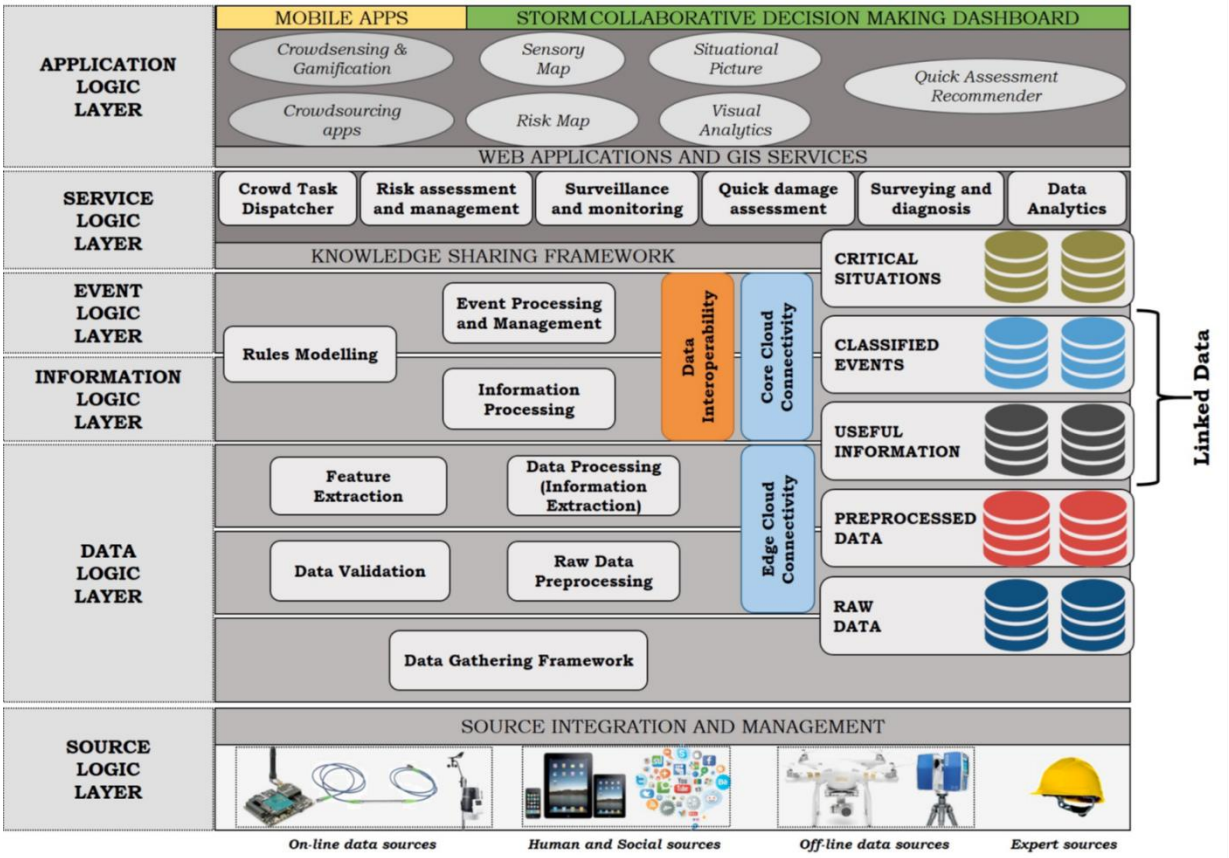

Figure 4. STORM platform's logical architecture

The *Source Logic Layer* is the lower layer and consists of online and offline data originated from IoT devices (e.g., sensors), human and social networks.

The *Data Logic Layer* contains functional modules that gather and process the data collected from heterogeneous information sources in the STORM system. All the pre-processed data (also those from offline, human and social sources) are then validated and elaborated again using data validation, processing, classification and enrichment techniques to produce useful information.

The *Information Logic Layer* contains information processing and fusion modules that deal with the processing of the useful information, produced by the data analysis modules, to extract classified events.

The *Event Logic Layer* defines a set of modules able to analyze and process (e.g., validate, aggregate and correlate) the classified events received as input from the Information Logic Layer, applying Complex Event Processing techniques.

The *Service Logic Layer* contains the service categories used by the STORM users to prevent, manage and mitigate the risk associated with natural hazards in the cultural heritage domain.

The *Application Logic Layer* allows for the users/experts to interact with the STORM services and tools using Web application technologies, GIS services and mobile apps for tablet and smartphone devices as well as crowdsourcing and gamification applications.

CLOUD-BASED INFRASTUCTURE

Cloud plays an important role in all the modern IoE systems, since it can host all the aforementioned services and is compliant with the non-functional requirements, such as scalability, availability and high maintenance. In this section we describe STORM's cloud-based infrastructure.

STORM Cloud architecture

STORM Cloud's architecture is based on a tree approach; it consists of one Core cloud and several Edge clouds. The Core cloud can be described as a brokering system while the edge clouds are cloud environments located near the sites of interest. The role of the STORM Edge cloud is the collection, storage and processing of data gathered by sensors. These data can be raw or pre-processed and are gathered by services that operate inside the virtual instances running on the Edge cloud. In order to achieve low latency, regarding data transferring, in network-constrained environments (e.g., Roman Ruins of Troia) Edge clouds should be located near each STORM site. However, if the site's workstations are not able to offer enough computational power for the support of an Edge cloud, then it can be served by the Edge cloud of a different site. Inside the Edge cloud, custom data processing services are deployed according to the needs of the site.

The STORM Core cloud is responsible for the collection of extracted information from the Edge clouds, the generation of events, the complex event processing, the communication with the STORM Edge clouds and the hosting of visualization tools and other services. Additionally, the Core cloud is also responsible for monitoring the status of the Edge clouds and their services that are running on virtual instances. Therefore, the Core cloud can be described as a brokering system, indicating which type of data each Edge cloud offers. Furthermore, several software services can be hosted also in the Core cloud which may interact with the Edge clouds and retrieve data and information related to the existing sensors. This type of communication between the Core cloud and the Edge clouds can be accomplished with the use of APIs (e.g., RESTful APIs some of which are described later in this section).

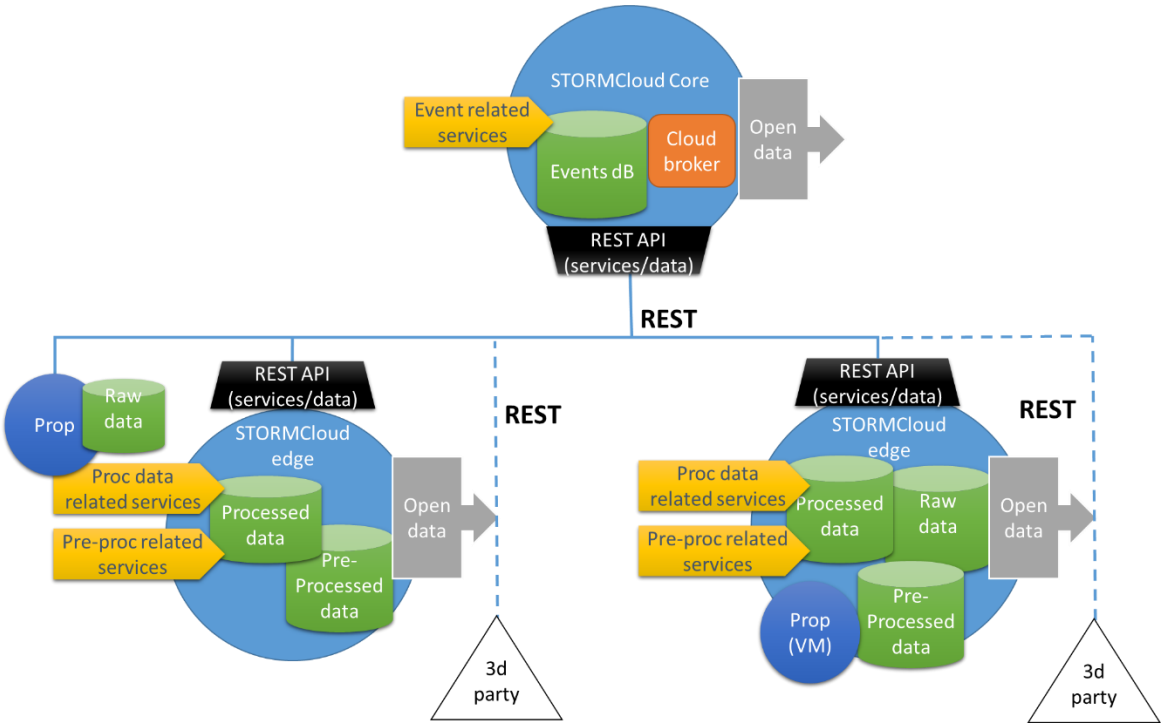

Figure 5. STORM Cloud Architecture

Figure 55 presents the cloud-based architecture followed in STORM. The STORM Core cloud communicates with each STORM Edge cloud through the use of REST APIs. The STORM Edge clouds are able to communicate with the Core cloud and with the other Edge clouds via APIs. Each STORM Edge cloud has its own data gathering mechanisms and stores raw, processed or pre-processed data. These data can be offered as Open data and are, also, used for the extraction of information which can be sent to the STORM Core cloud. Then, the STORM Core cloud will store and process the information and produce the events using the required services and tools.

Additionally, the Core cloud hosts services which are able to communicate with the Edge clouds via the use of APIs and present information related to data and state of specific sensors in several sites. Finally, as it was mentioned earlier, the STORM Core cloud is able to act as a brokering system for the data sharing between the Edge clouds.

INTRA-CLOUD COMMUNICATION

One of the key functionalities of the STORM cloud infrastructure architecture is the communication between:

- The Sensors and the Edge clouds
- The Core and the Edge clouds
- Inside the Core cloud (intra-Core communication)
- The Edge clouds
- The Edge clouds and third-party providers

The aforementioned communications can are fulfilled by taking advantage of the STORM cloud components:

- Edge Cloud Connector
- Core Cloud Connector
- Cloud Broker

which are based on RESTful services (e.g., communication between Core and Edge cloud) and the Publish-Subscribe pattern (e.g., communication between Edge and Core cloud or for intra-Core cloud communication).

The RESTful services are triggered when a REST call has been initiated. It is worth mentioning that the server - client model is followed in this technology, nevertheless the server is not concerned about the status of the client and vice versa. Therefore, the RESTful services are characterized as stateless. For example, if a REST call is performed by the client and the server status is down then the server will not answer and the client should perform another call later and, if the status is up, the server will respond.

On the other hand, the Publish – Subscribe (Pub/Sub) pattern is ideal for instant and asynchronous messaging. In this messaging pattern, the senders of the messages are called publishers (producers) and the receivers of the messages are called subscribers (consumers). Moreover, the publishers do not send the messages directly to the subscribers, but, instead, they categorize the messages into classes called topics. Consequently, the consumers subscribe to the topics of their interest (i.e., topic-based filtering) in order to acquire the information provided by the publisher. The usage of the Pub/Sub technique increases the scalability of the system and enables real-time notifications.

Figure 66 illustrates the STORM cloud infrastructure and communication architecture, focusing on the communication flows between the STORM clouds. In particular, the sources (e.g., sensors) produce data that are collected by the Data Gathering Framework and are validated or/and preprocessed by the relevant components, which are deployed to the Edge cloud. Afterwards, the structured data are sent to the Edge Cloud Connector (ECC) and to the Data Processing components. The ECC is a RESTful service that stores the real-time data to a NoSQL database and sends them (e.g., the last measurement of a sensor) directly to the Core Cloud Connector (CCC), as well as the Data Processing components extracting useful information from the structured data.

Since CCC is based on the Pub/Sub pattern, he acts as a broker and offers the extracted information and the real-time data via topics. In case of information, the subscribed Information Processing component generates from them classified events, which can be stored to the relevant repository or can be sent directly to the Event Processing components and to the CCC. Moreover, the Event Processing component that receives the generated events apply Complex Event Processing techniques to identify the correlated events (in terms of time and space), which can be, also, stored to a database or sent to the CCC. Finally, STORM services, tools, and applications have the ability to receive all kind of historical or real-time data (structured data/information/event/correlated events) utilizing REST APIs or the Pub/Sub pattern.

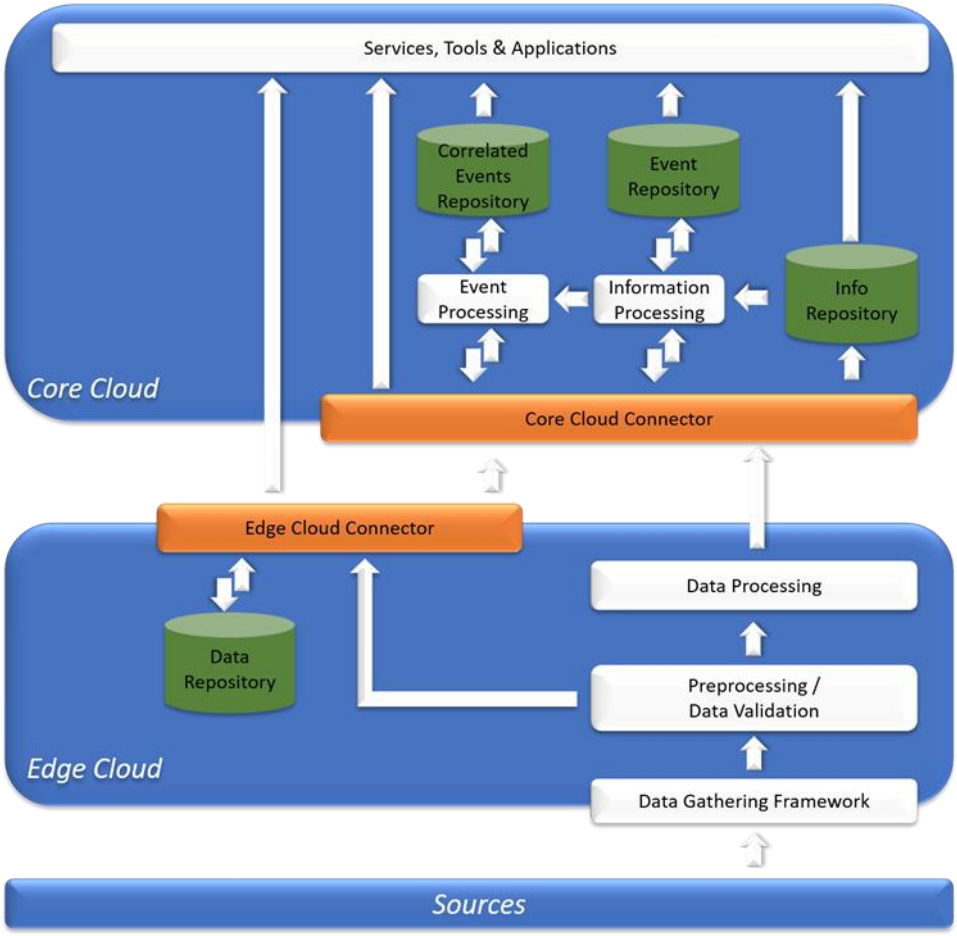

Figure 6. STORM cloud infrastructure

The following subsections explain in detail the mechanisms of the ECC and the CCC, and introduce the Cloud Broker, which is responsible for monitoring the cloud resources.

Edge Cloud Connector

The ECC component is a RESTful webservice that is responsible for handling the structured data and runs at the STORM Edge cloud. In STORM, there can be two kinds of structured data: the data regarding the last measurement of a sensor (i.e., almost real-time data) and historical data (i.e., values that might cover a longer period of time in the past). The former are forwarded to the CCC, while the latter are stored in a NoSQL database (e.g., MongoDB).

Figure **7**7 illustrates the RESTful interactions of the ECC. In particular, the ECC receives via a POST method (e.g., http://{DomainName}/storm/rest/manage/edgecloudconnector/data) by the Data Validator / Preprocessor components all the incoming structured data and stores them to the NoSQL database. These data can be offered, afterwards, to the STORM consumers. The initiator of the communication should define the kind of data that is expecting, by making the right calls (e.g., http://{DomainName}/storm/rest/manage/edgecloudconnector/site/{site}/nodes/sensors/list) and define the time period of the requested historical data (e.g.,

http://{DomainName}/storm/rest/manage/edgecloudconnector/nodes/{node}/sensor/{sensor}/data/date/{start}/{end}/limit/{limit}).

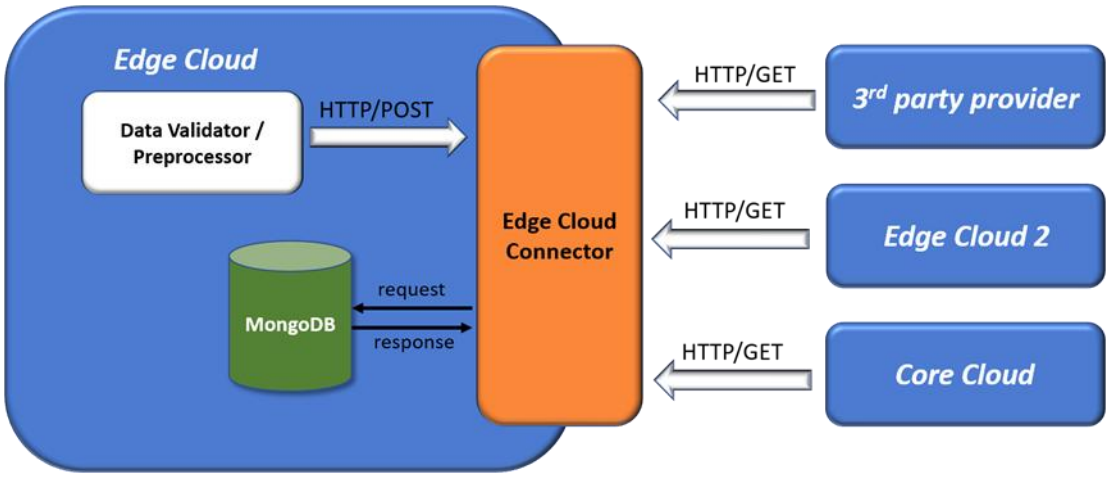

Figure 7. Abstract representation of the RESTful service offered by the Edge Cloud Connector

There are three kinds of entities that consume the services of the ECC, 1) the Core cloud, 2) the Edge clouds, and 3) third-party providers, which initiate the following communication flows:

1. Core to Edge cloud communication

This communication flow deals with the transfer of data between RESTful services hosted in an Edge cloud and applications or services running at the Core cloud. The transferred data are retrieved, by the RESTful services that are running at the Edge cloud, from the existing structured data repository. Then, they are forwarded to the service or application at the Core cloud that initiated the communication.

2. Edge to Edge cloud communication

As depicted in Figure **7**5, the communication between the Edge clouds will, also, be achieved through the use of the RESTful services. For example, if a data processing service deployed to an Edge cloud needs data stored in another Edge cloud, then it will have to make an HTTP request to the ECC, to retrieve these data.

3. Third-party providers to Edge cloud communication

For the communication between a third-party provider and an Edge cloud (preferably in the form of open data) REST APIs will be used, once more.

It should be noted that in any of the aforementioned cases the consumers will be firstly authenticated, in order to get the required API token from the ECC.

Core Cloud Connector

The transfer of real-time data and useful information from the Data Processing components to the CCC is based on the Pub/Sub pattern. Figure **8**8 presents the communication between the Edge cloud and the Core cloud, and the intra-Core cloud communication. The proposed mechanism provides many-to-many,

real-time, asynchronous messaging decoupling senders and receivers. Moreover, it allows for secure and highly available communication between independently written applications. In this way, notifications will be triggered at the subscribed to the topic Core cloud services (e.g., the Event Processor) and/ or applications about any updates, originating from the Edge clouds.

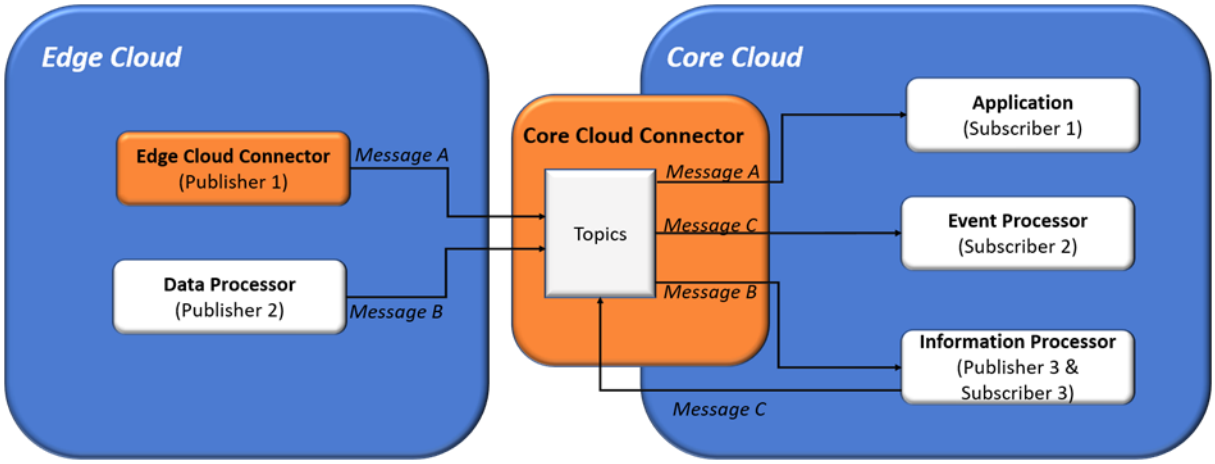

Figure 8. The Pub/Sub mechanism of the Core Cloud Connector component

1. Edge to Core cloud communication

The CCC receives messages from the ECC containing real-time data (i.e., last measurement) originated by sensors and from the Data Processing components containing useful information. The messages will be using a JSON format. An example of such a data format is given in Figure 9.

2. Intra-Core communication

In the Core cloud, the Pub/Sub method is, also, used by the CCC for the sharing of event/situation between the hosted components (i.e., Event Processor, services and applications). This kind of messages will, also, be using a JSON format.

```
[ { "type": "AirTemperature",
    "topic": "AirTempBoD1",
    "unitMeasurement": "Celsius",
    "value": 24,
    "nodeId": "eiciji43424i",
    "datetime": {
      "created": "2018-02-19 13:40:23.234",
      "transmited": "2018-02-19 13:40:23.334",
      "received": "2018-02-19 13:40:24.334"
    },
    "site": {
      "siteName": "Baths of Diocletian",
      "heritageAsset": "Michelangelo's Cloister"
    },
    "location": {
      "point": {"latitude": 41.321, "longitude": 12.4984
      }
    },
    "status": 1
  }
]
```


Figure 9. Example of the information message in JSON format

Cloud Broker

The Cloud Broker is a component hosted inside the Core cloud, which provides a directory service for services and a dashboard for monitoring the resources of STORM clouds. In addition, when a new service or new dataset is available through the STORM Cloud REST API, a notification to the Broker is provided through the use of CCC, so that, at each time, the Broker is aware of all the available services and data at the cloud instance (Figure **10**10).

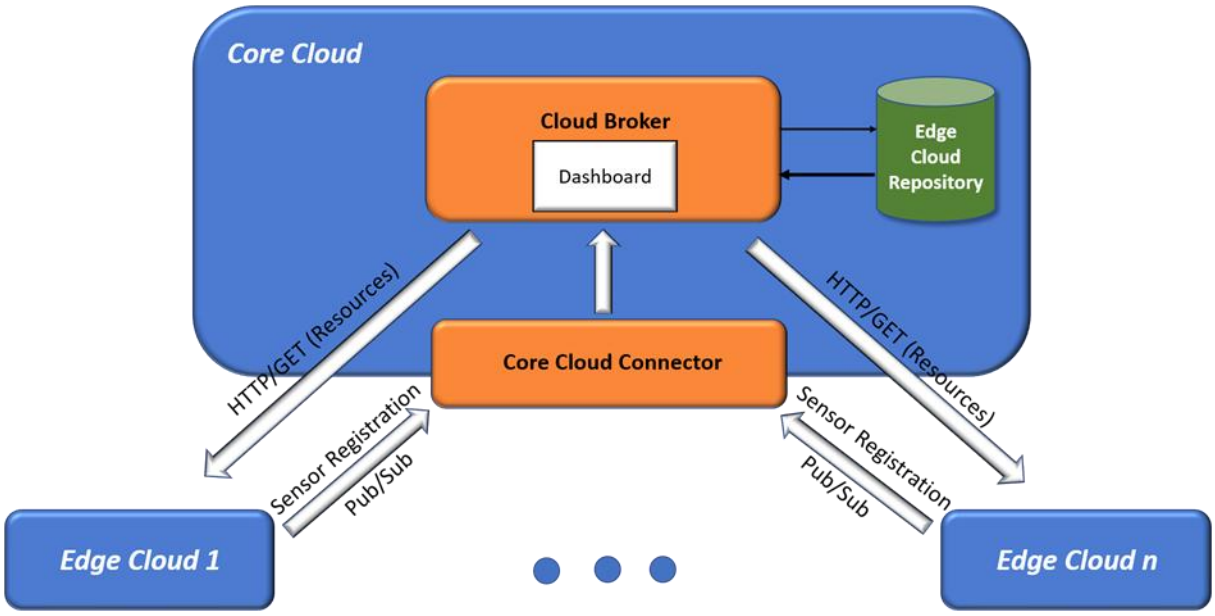

Figure 10. Representation of the Cloud Broker component

The interface of the Cloud Broker component was specified to interact with the Edge clouds through the usage of OpenStack RESTful APIs (Openstack, 2018). The RESTful services (authentication and monitoring) are described in Table 4.

Table 4. Cloud Broker's RESTful API

Method	Explanation
cloudBrokerAuth	The registered Edge cloud owners are authorized by the Core cloud.
os-simple-tenant-usage/{tenant_id} (OpenStack API 2018)	The Cloud Broker will consume the webservices of the OpenStack Edge clouds in order to get updated about their status.
requestData	Each component running at the Core cloud will receive data using the methods of ECC and CCC as mentioned before.

FUTURE ENHACEMENTS

One of the goals of the STORM project was to be considered as a source of information by Cultural Heritage experts. The information that could be shared might include suggestions about the methodologies that can be applied while dealing with disastrous events in a Cultural Heritage site, along with details about the deployment of state of the art equipment and the implementation of a system architecture that is robust, resilient and efficient.

To this end, one of the possible enhancements of the proposed IoE system, that is produced for STORM project, is to include the notion of the *Digital Twin*, meaning the virtual identical sibling of a natural exhibit

or artifact, that is located on a Cultural Heritage site. This virtual sibling can be used, under the STORM logic, to provide for information regarding the effects (detrimental or not) that certain natural or human-related conditions have to the artifacts health and condition. At the same time, the digital twin can take part at simulations, where the conditions might change to simulate either the transfer of the physical object to a different environment or the behavior of similar objects that can be found in different sites. Through the simulations, possibly deployed on the cloud or at the edge using fog computing and sharing of the results, the experts can share details about their findings and contribute strongly on the preservation techniques that need to apply on different situation based on the (natural) conditions while, also, describe the warnings that could trigger the alarms for the prevention of detrimental actions to the exhibits.

Finally, having in mind the notion of the digital twin that was described above, an expansion to the provided infrastructure is, also, expected. This expansion will be focusing on the use of (virtual) containers as a means to serve the described simulations and to contribute on the overal minimization of the risks by cyber-attacks. Containers are more and more considered as a very good technique that can help with increasing the system's performance, while managing the available resources in a more efficient (both regarding the performance and the energy consumption) way and can play a significant role in the described IoE architecture.

CONCLUSION

In this article, the use of IoE and cloud technologies for enabling efficient protection and preservation of Cultural Heritage sites and artefacts, having in mind the need for using low cost solutions, easy to be deployed. The described framework can be considered as the basis over which future and emerging concepts, combining the results in different domains where IoE can be applied (i.e. smart cities), may be exploited. The use of a scalable, easy to deploy framework as the one presented here, making use of heterogeneous infrastructures can provide the ground over which successful concepts such as the Digital Twin (a virtual representation of a tangible item, enabled through APIs which introduce the concept of servitization rather than the traditional digitization) can be introduced. Links to solutions enabling intelligence towards the improvement of quality of life in cities can further enhance the benefits from the use of IoE and cloud technologies, creating new opportunities for the economy and tourism, through the parallel exploitation (with respect to the preservation) of Cultural Heritage.

ACKNOWLEDGMENT

Work presented in this paper has received funding from the European Union's Horizon 2020 research and innovation programme under STORM project, grant agreement n°700191.

REFERENCES

Chianese A., Piccialli F., Jung J. E., (2016). "The Internet of Cultural Things: Towards a Smart Cultural Heritage", in the *12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*.

Chianese A., Piccialli F., (2015)."Improving User Experience of Cultural Environment Through IoT: The Beauty or the Truth Case Study", in Intelligent Interactive Multimedia Systems and Services. Smart Innovation, Systems and Technologies, vol 40. Springer, Cham.

Chianese A., Marulli F., Moscato V. and Piccialli F., (2013). “Smartweet: A location-based smart application for exhibits and museums,” International Conference on Signal Image Technology and Internet Based Systems, pp. 408–415.

IEEE, (1998). "IEEE Std 830: Recommended Practice for Software Requirements Specifications".

Minerva R., (2014), “From Internet of Things to the Virtual Continuum: An architectural view”, in IEEE Euro Med Telco Conference (EMTC), 12-15 November.

Openstack (2018), OpenStack Installation Guide for Ubuntu, URL: https://docs.openstack.org/mitaka/install-guide-ubuntu/, last accessed 31/01/2018.

Ott M. and Pozzi F., (2011). “Towards a new era for Cultural Heritage Education: Discussing the role of ICT”, in Computers in Human Behavior, vol. 27, issue 4, pp.1365-1371. 10.1016/j.chb.2010.07.031.

Perles A., Pérez-Marín E., Mercado R., Segrelles J. D., Blanquer I., Zarzo M., Garcia-Diego F. J., (2018). “An energy-efficient internet of things (IoT) architecture for preventive conservation of cultural heritage”, in Future Generation Computer Systems, Volume 81, pp. 566-581.

Sigfox, (2017). SIGFOX - The Global Communications Service provider for the Internet of Things (IoT), URL https://www.sigfox.com/en, last accessed May 2018.