Personal and Ubiquitous Computing (2023) 27:379–399
https://doi.org/10.1007/s00779-022-01698-6

ORIGINAL PAPER

Enhancing user interaction with context-awareness in cultural spaces

Konstantinos Michalakis[1] · George Caridakis[1]

Received: 29 December 2021 / Accepted: 18 October 2022 / Published online: 2 December 2022

Abstract

The emergence of the Internet of Things has fueled a proliferation of smart things in many fields, including cultural spaces. Context-awareness addresses the production of large volumes of context by analyzing raw data and adding a meaning to them. Middleware systems have emerged, which perform context modelling and reasoning, supporting context-aware applications. The services provided by such applications can be personalized, automated and adapted to the current situation, thus enhancing the user interaction with the devices and the digital environment. In this work, a context-aware middleware system is presented, based on a hybrid reasoning schema, which combines multiple techniques to efficiently address each problem. The proposed middleware system is evaluated in a cultural space, where scenarios were designed and tested, using a mixture of real and artificial data. The experiments measured the accuracy, performance in terms of reaction time and scalability and the interactivity enhancement, achieved by the proposed middleware.

Keywords Context-awareness · Intelligent interaction · Internet of Things · Middleware

1 Introduction

Nowadays, we have grown quite dependent on computing devices for the realization of various tasks. Surrounded by smart things, people interact with them in their daily activities. In addition, smart devices are functioning independently of users' actions, collecting data, processing and acting upon them, either anticipating users' needs or serving machine-to-machine purposes [1]. Ubiquitous computing realizes the connectivity of many smart devices that offer personalized or automated services to the users [2]. Utilizing the existing networking infrastructure, the Internet of Things (IoT) paradigm promises connectivity for anything, anywhere and anytime, transcending the limits of closed ecosystems and covering all the world [3].

The proliferation of sensors and smart devices along with the emergence of numerous IoT applications will result in the generation of huge volumes of data. Yano et al. [4] argue that "big data without link to value is merely a cost". Context-aware computing addresses this problem by analyzing raw data and adding a meaning to them [5]. Context-awareness (CA) is the ability of the application to be aware of the current situation and act accordingly [6]. A CA application can enhance the human-computer interaction by identifying the user's current needs, preferences and limitations. Furthermore, machine-to-machine interaction can also be improved by applying context-awareness. In general, most IoT applications incorporate context-awareness [7].

The IoT paradigm envisions a large number of connected smart devices, sensors and other objects. The heterogeneity of deployed objects, the reusability of produced context and the limitations in computational capabilities have led to the integration of middleware. Such systems are integrated in IoT ecosystems in order to address those issues, connect with varying context sources and efficiently disseminate the produced context to interested parties [7]. Middleware

Konstantinos Michalakis and George Caridakis contributed equally to this work.

✉ Konstantinos Michalakis
kmichalak@aegean.gr

✉ George Caridakis
gcari@aegean.gr

[1] Cultural Technology and Communication, University of Aegean, University Hill, Mytilene, 81100, Greece

systems abstract communication details of smart devices and provide interoperability and diversity for applications and services [8].

Context [5] needs to be modelled and reasoned, before disseminated to context consumers. Context modelling includes techniques such as key-value, markup schemes, graphical models, object-based models and ontologies [9]. Reviewing the proposed techniques, Perera et al. [5] argue that the incorporation of multiple modelling techniques in the same middleware allows the mitigation of each other's weaknesses. On the other hand, context reasoning may be performed using probabilistic logic, rule-based logic, fuzzy logic, supervised and unsupervised learning or ontology-based logic [10]. Since reasoning techniques are highly correlated to the problem in hand, the selection of the appropriate technique is usually dependent on the application's needs. Lately, hybrid solutions have emerged to formulate complementary techniques in order to address each other's weaknesses [11].

Cultural spaces nowadays, endeavor to attract visitors by exploiting new technologies and offering new and exciting ways to promote cultural content and activities. Digital guides, interactive screens, personalized applications engage cultural visitors offering a new type of interactive user experience [12]. Smart cultural institutions have emerged with the integration of IoT installations. Typically, visitors use their mobile devices to traverse the cultural space and interact with artifacts and other users. Recommendations and personalized content are tailored to the user profile and the current situation, while the environment is customized accordingly [13].

The current work proposes a context-aware middleware which aims to enhance the interactivity of visitors in a cultural space. The middleware integrates context acquisition, modelling, reasoning and dissemination on the server, supporting context-aware applications which act as context consumers. The context modelling technique used by the middleware is based on a model proposed in [14] which utilizes five core classes: thing, activity, location, time and asserted context. The proposed context reasoning technique used by the middleware is a hybrid approach which exploits rules, probabilistic logic and supervised and unsupervised learning in order to address each specific problem with the most suitable technique.

The proposed middleware was evaluated in a cultural space, utilizing a mixed dataset which consists of existing profiled data of users and artifacts, artificial behavioral data of users and actual environmental data. The middleware was evaluated under specific scenarios that created artificial situations and measured the system's response to them, by comparing the expected with the actual contextual output. Furthermore, the middleware was tested in terms of performance and scalability. Overall, the system showed high accuracy in producing the expected context and also featured quick reaction times and ability to scale well with increasing numbers of connected objects.

The contribution of the current research work is threefold. Firstly, a novel context reasoning schema is proposed, which exploits hybrid reasoning and the blending of real-time and on-demand reasoning in order to capture the dynamics of a fluid ecosystem of a cultural space. Furthermore, the categories of context-aware services are identified and analyzed. Secondly, the evaluation method utilizes a blending of real and artificial data in simulated experiments, allowing the evaluation of performance and scalability as well as the efficiency of the system in extreme situations, indices which are harder to test in real scenarios. Lastly, the proposed middleware explores the enhancement of user interaction within cultural spaces which are enriched with context-awareness, allowing for more personalized user experiences.

The paper proceeds as follows: Section 2 presents an overview of related work with regard to context-aware middleware that support generic and cultural environments. Section 3 describes the proposed context-aware middleware in detail, including the architecture, the context model, the context reasoning procedure and the context-aware services provided. The experimental setup is described in Section 4, including the dataset and the scenarios included. Section 5 describes the validation process, with analysis of the infrastructure and implementation. The results of the experiment are analyzed in Section 6. Conclusions and future work are discussed in Section 7.

2 Related work

Visitor experience is defined in [15] as "an individual's immediate or ongoing, subjective and personal response to an activity, setting or event outside of their usual environment". A definition of the cultural user experience has been the focus of our earlier work [16]. Lately, a discourse fueled by postdigitalism, investigates how museums can include digital media in a normative way [17]. Instead of mutual exclusion, Parry suggests a blending between digital and non-digital elements, augmenting each other, while keeping digital sources ambient and aesthetically acceptable. [18] proposes a conceptual tool and suggests specific design elements that incorporate the post-digital museum experience, consisting three elements: experience, structure and surface. Visitors are also perceived as acquiring new roles such as co-creators of their own cultural experience [19].

In our previous work [20] we have reviewed the contextual elements of a cultural visit. Influenced by [21], the survey proposes a set of six context types such as social, interactive or environmental which capture the various characteristics of the visitor, according to related studies on the museum experience [22–25]. The current work moves the focus from visitor experience to implementing the proposed contextual schema and evaluating the enhancement of user interaction achieved, while following the principles of the post-digital museum and the ambient intelligence offered by CA.

Generic context-aware platforms have been widely researched the past years, due to the emergence of the IoT paradigm. A detailed survey of context-aware middleware is presented in [5, 7]. Among the first context-aware middleware, CoBrA [26] focuses on smart meeting places. It tackles the resource constraint characteristic of sensors and features a context broker that allows personalization. The centralized system allows distributed brokers that function as a federation and are linked in a single smart space. Hydra [27] functions as an IoT middleware that supports context acquisition, management and interpretation. The integrated rules engine receives data sources about artifacts, semantic context and application content.

SeCoMan [28] uses the Semantic Web to model context, to reason over data to infer new knowledge, and define context-aware policies. It employs three actors: framework administrator, application administrator and users. It utilizes a location ontology which is categorized into element, authorization and space. ACAIOT [29] is a semantic-based and domain-independent framework that supports context-aware applications. It enables an abstraction for adaptable services to multiple domains. The architecture consists of the context management layer, the library and the service layer. Context modelling includes pre-processing, aggregation, and modelling and applies an ontology. The rules engine executes two types of rules: activity rules and service rules.

Context-aware platforms designed for cultural spaces have also emerged in the past decade. The most prominent ones, in terms of interactivity enhancement and middleware integration, are presented below. The platform described in [30, 31] utilizes a gateway server in order to host the processes that acquire context from mobile devices. The gateway is responsible for the acquisition of content from the content server as well. The platform exploits implicit and explicit user profiling and delivers content according to the fused profile. The work has been expanded in [32] by adding a context manager.

IRME [33] classifies users according to their actions and profile. Grouping visitors, the platform invites them to share needs and experiences accordingly. This work integrates middleware functionality by transferring the work flow to the Cloud. The applications are connected to the sensory layer and the cloud middleware tackles management of digital material. The platform abstracts content from various sources into a single information entity.

The context-aware assistant for cultural environments SCRABS [34] abstracts the sensor network as a distributed database, accessed with SQL-like queries. The implementation relies on wrappers that are specialized source objects, such as a sensor node, addressing the interoperability issues and providing seamless access to heterogeneous data sources. SCRABS exploits both static and dynamic information to define user profiles. It also maintains a dynamic profile for the user, based on their behavior such as movement and actions. Finally, it uses RFID technology to identify the proximity of a visitor with an object of interest.

The multi-protocol middleware proposed in [35, 36] does not exploit any profiled data but exploits only location data for the delivery of appropriate content to the visitors. Analysis of visitor behavior allows the proximity algorithm to identify wrong artwork placement or unexpected user movements. It allows transparent access to heterogeneous IoT protocols, providing scalability and flexibility as well as the ability to extend to new technologies, while currently supporting CoAP, KNX and BLE communication protocols. Finally, it provides services to the processing center of the smart museum through high-level RESTful APIs.

The framework described in [37] integrates a middleware which is responsible for the collection and management of sensory messages. The middleware executes pre-filtering of context data and also includes a recommender engine. It deduces user preferences based on their selection of interesting artifacts, while also maintaining a list of their ratings. Typical GPS localization without taking into consideration profiled data is also performed.

Finally, the meSch project [38] gathers user profile data at registration time, including explicit user preferences about their cultural visit. It allows the localization of users through proximity with a beacon (supporting either BLE or NFC connectivity), by abstracting the beacon devices within the meSch ecosystem and integrating them in preconfigured clusters of sensors. The reaction of the system to user movement is programmed to enable script variations to the story, such as different sounds depending on the visitor's movement towards a PoI.

3 Middleware

3.1 Architecture

The proposed context-aware middleware follows the optimal context life-cycle introduced in [5]. Specifically, context is manipulated in four distinct stages: it is acquired from various sources, it is modelled and stored accordingly, it is reasoned and finally it is disseminated to interested stakeholders. Those four stages are depicted in Fig. 1, accompanied by other internal or external components. The

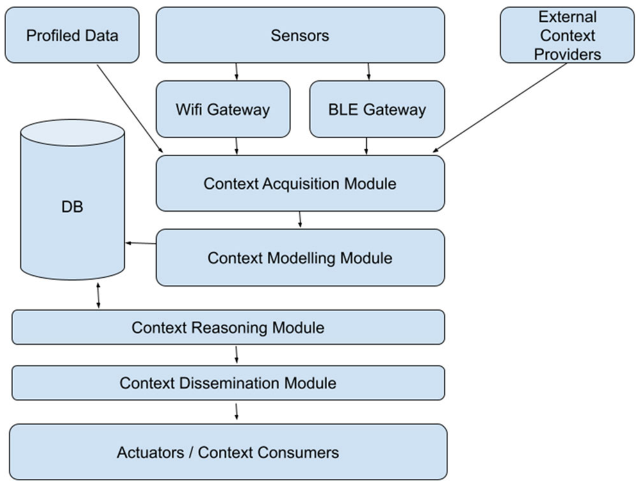

Profiled Data
Sensors
External Context Providers
Wifi Gateway
BLE Gateway
DB
Context Acquisition Module
Context Modelling Module
Context Reasoning Module
Context Dissemination Module
Actuators / Context Consumers

Fig. 1 Middleware architecture schema

context categorization schema proposed in [39] and adapted in [14] includes sensed, profiled and derived context. This categorization is depicted in the proposed architecture as follows:

- Profiled data are explicitly illustrated as an independent context source. Such data usually are stored either in the system DB or the application DB. In the first case, context acquisition is trivial, while in the latter, profile retrieval functionality is necessary. Usually, proprietary profiles stored in applications are less efficient due to lack of standardization and interoperability. Less often, profiled data may also be retrieved by external context providers.
- Sensed data have multiple sources: sensors installed in the area, mobile devices with equipped sensors carried by users, user input which is provided through public or private devices and finally external context providers such as weather or traffic conditions. Figure 1 distinguishes two types of context sources: internal named Sensors and external named External context providers.
- Derived data are not depicted in the architecture. Such context is not created at the acquisition stage, but rather at the reasoning stage, when appropriate higher level context is required. Thus, from a procedural view, derived data are not illustrated in Fig. 1.

Sensed data, i.e., data that are not stored in any type of DB, need to be captured by the middleware, through a network protocol. The proposed architecture supports WiFi and BLE connectivity. WiFi is the leading protocol for wireless communications, which is highly compatible with most modern devices. On the other hand, Bluetooth Low Energy (BLE) devices are very common in many IoT installations, offering low-consumption functionality and consistent connectivity. Both popular protocols are supported by the Context Acquisition Module which is responsible for collecting context from various sources, offering interoperability. The expansion of the proposed architecture with more protocols (such as Zigbee, another IoT favored protocol) is easily performed, satisfying a possible demand for such compatibility by a new installation.

The above categorization is based on functional criteria. A second categorization schema is based on the actual cultural visit and incorporates the findings of [20]. There are six categories of contextual data based on the cultural visit: (a) stable visitor profile, e.g., age and background, (b) context related to current visit, e.g., motivation, attention level and fatigue, (c) interaction context, e.g., location and proximity, (d) social comments, e.g., communication within groups and shared experiences, (e) environmental context, e.g., temperature and (f) context of content, e.g., narrative methods. Typically, (a) is profiled, while (b)–(f) are either sensed or derived data, but the purpose of the second categorization schema is not to identify sources of context but rather consumers of high-level context.

The rest of the modules are straightforward. Context acquired with the Context Acquisition Module is stored in the DB by the Context Modelling Module after it has been modelled according to the schema described in Section 3.2. The Context Reasoning Module is connected with the DB with a bidirectional arrow, which indicates that it both retrieves context and stores new context (of higher level) to the DB. The reasoning functionality and data flow are described in more detail in the next paragraph. The Context Dissemination Module acquires context from the Context Reasoning Module and disseminates it to actuators and context consumers.

3.2 Context modelling

The proposed middleware implements a context modelling technique presented in [14] The abstract modelling schema is illustrated in Fig. 2, which depicts the five main core classes. It is based on ontology modelling, borrowing the representation capabilities of ontologies to efficiently capture context in IoT ecosystems.

At the core of the context model lies the thing, the building block of the IoT. The concept of things in this model includes all active objects, i.e., devices that have networking capabilities, but also people connected to the network through their wearable and mobile devices or even through activity. The location class allows spatial relations among things. The activity class represents activities that are performed by things and is wide enough to cover user activities but also device activities and status changes. The time class is connected to the activity and allows temporal

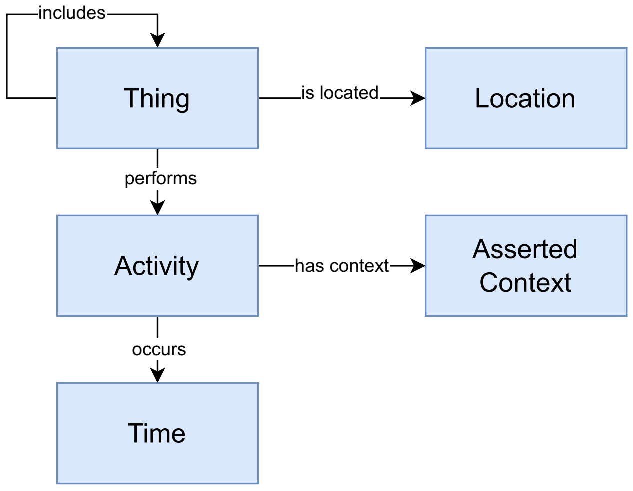


Fig. 2 Context modelling schema

specification of events. The asserted context class is a novel addition which allows the modelling of context that is related to a certain activity, e.g., the turning on of an air conditioning device which is related to context such as room temperature and user profile.

Each one of the five core classes includes sub-classes that model characteristics such as profile and role. A more detailed analysis of the implemented context model is available in [14]. Mapping the core classes to the cultural space objects, Thing can be considered a sensor attached to an exhibit, a sensor related to functionality or a visitor. Location is the position of visitors and exhibits in the room, also related to other exhibits or visitors. Activity is related to visitor's actions related to exhibits such as interacting with a screen next to an exhibit or adjusting the settings of a digital media device. Asserted context is the context derived by the system that describes the visitor's interaction, structured in machine language.

3.3 Context reasoning

The proposed context-aware middleware exploits a hybrid reasoning schema in order to efficiently support multiple types of context. Many different techniques have been proposed in the literature, each one with its own strengths and weaknesses. The most commonly used techniques are rules (either accompanying an ontology or independent), machine learning techniques (supervised and unsupervised learning), probabilistic logic and fuzzy logic [10]. Each technique excels in different types of context and complex context representation requires that the appropriate technique should be used in every scenario.

Applying this concept, a hybrid reasoning schema is presented in Fig. 3, which focuses on the last three components of the larger architecture of Fig. 1. The proposed

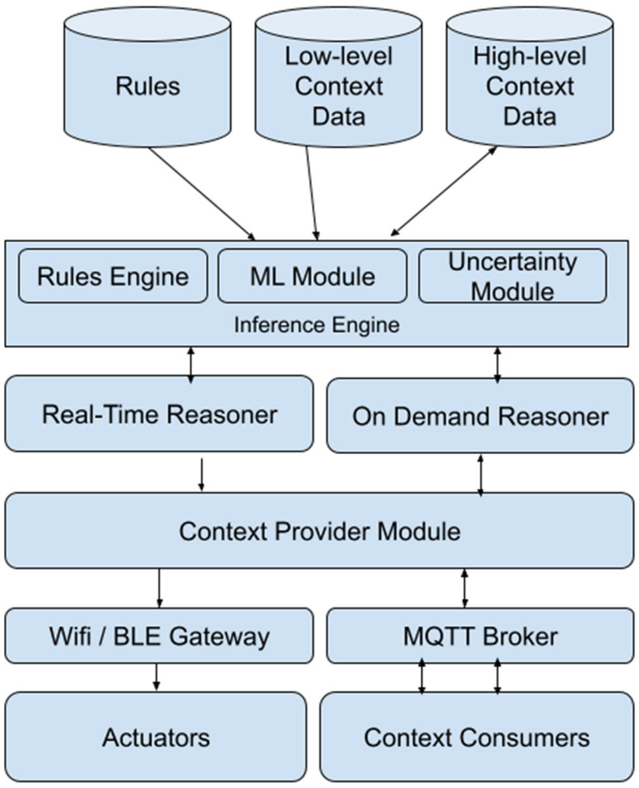


Fig. 3 Context reasoning schema

context reasoning schema is based on three premises: two levels of context, hybrid reasoning and real-time and on-demand reasoning.

Two levels of context Context is dynamic in nature and highly hierarchical. The lowest levels of context include only raw measured values from sensors. The highest levels of context include information such as which activity is performed by the user and with whom. Between those two extremes, context of varying expressiveness is collected, modelled, stored and fused. In the proposed reasoning schema, the distinction is between low-level context that includes raw measurements and low-complexity processing (such as the calculation of the distance between two objects) and high-level context that includes derived contextual information of higher detail. A typical reasoning cycle corresponds to the processing of low-level context into high-level context. Thus, context stored in the DB is always dynamic, changing according to the latest contextual data acquired by the context sources.

Hybrid reasoning As indicated in the literature [5, 11], hybrid reasoning is necessary to tackle real-world situations where different context types are required. Figure 3 indicates three modules that perform context reasoning, the combination of which is named Inference Engine.

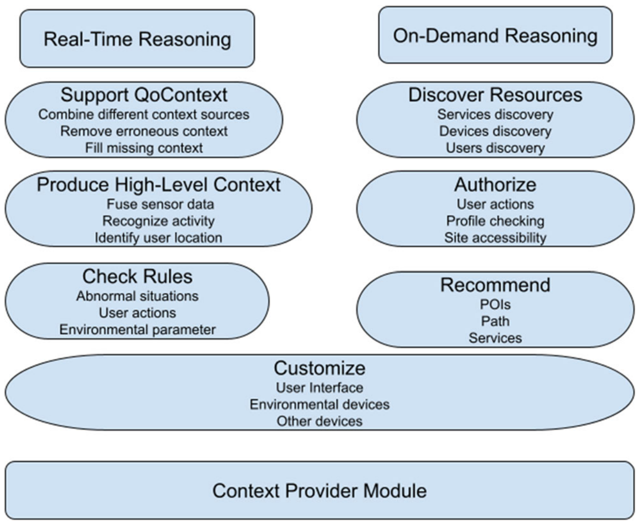


Fig. 4 Context reasoning services

The Rules Engine is responsible for performing reasoning that is based on rules, which may be accompanied by ontologies or other types of context modelling, such as graphical models. The majority of reasoning is performed by the Rules Engine, since this technique is appropriate to tackle generic problems. The Machine Learning (ML) Module is responsible to perform reasoning for appropriate problems such as activity recognition and identification of abnormal measurements. Those problems are very complex to be addressed by traditional techniques and are tackled by supervised and unsupervised learning methods. Finally, contextual data inherit uncertainty due to noise, incompleteness and inconsistency [40]. The Uncertainty Module mitigates the effects of uncertainty on context reasoning by exploiting probabilistic logic. Each module is executed independently of the other, according to the needs of the current situation.

Real-time and on-demand reasoning According to another characterization schema [10], context is either event driven or time driven. Expanding upon this schema, Michalakis et al. [14] proposed that context may also be on-demand. Following the above categorization, the proposed middleware supports real-time reasoning and on-demand reasoning. Real-time reasoning refers to both event driven and time driven context. Either periodically, or triggered by an event, real-time reasoning is continuously executed and calls the inference engine in order to extract the required derived context. On-demand reasoning, on the other hand, is triggered after a user or application request. Their main difference lies in the source of context request, which is internal for real-time reasoning and external for on-demand reasoning. This differentiation also ensures the distinction between middleware and application in respect to context needs, in other words, the need of the system to feature consistent context-awareness versus the need of the application to adapt to the user's needs.

The Context Provider Module is part of the Dissemination Module and is responsible for the acquisition of either context requested by the application/user or context produced by automated functionality. The produced context is disseminated to two different types of endpoints: (a) the actuators which act upon specific context, such as turning on an air conditioning device. Similar to sensors, actuators may be connected using WiFi or BLE protocols. (b) The context consumers which are typically applications requesting context in order to adapt, customize, recommend. A Message Queuing Telemetry Transport (MQTT) Broker is exploited in order to provide a light-weight publish-subscribe protocol for IoT messaging.

3.3.1 Context reasoning services

As previously mentioned, there are two types of reasoning: real-time and on-demand. Each one of them supports different context-aware services, which are illustrated in Fig. 4 and will be described next.

Real-time Reasoning

- *Support Quality of Context*. Quality of context has long been recognized as an important aspect of context-aware applications [41]. It refers to the procedures that ensure that the acquired context has sufficient quality to support context-aware applications. Context is imprecise, erroneous and incomplete. Various methods help to mitigate those inconsistencies. More specifically, the proposed middleware combines different context sources taking into consideration source reliability and history, removes erroneous context, accordingly notifying the source of that context and fills missing context, based on historic values and other parameters. Probabilistic logic is exploited in most actions performed under this category.
- *Produce High-Level Context*. Most context consumers require high-level contextual information which is more meaningful than unprocessed context data. The fusion of sensory data allows the production of generic high-level context, but also more specific processes are provided such as recognition of user activity and identification of user location. Such processes are continuously executed in order to ensure that the system has an up-to-date version of the current situation, in respect to user behavior. High-level context is important in

cultural spaces since human activity is complex and requires higher levels of abstraction in order to reach to safer conclusions on what are the needs of the visitor currently in terms of content delivery.

- *Check Rules*. Checking the conditions of rules and performing the equivalent actions in case of satisfied conditions, is probably the most straightforward context-aware service. Typical rules include the identification of abnormal situations, the reaction to changed environmental parameters and the reaction to user actions. Although rule checking is not considered computationally expensive, a possible large number of rules that need to be checked periodically may reduce the system performance significantly. Rules in cultural spaces are seldom user-oriented, since the visitor usually lacks the skills or will to setup their own preferences, thus most rules are computer-oriented and tackle automated functionalities such as environmental control and authorizations.
- *Customize*. Customization refers to the adaptation of the user's environment (including the user interfaces of devices) according to their needs. Such processing may be triggered automatically, during real-time reasoning or manually, when a user interacts with the context-aware application. Thus, the customization service is shared to both real-time and on-demand reasoning.

On-Demand Reasoning

- *Discover Resources*. Discovering nearby resources is a constant requirement for users of digitally enhanced ecosystems such as a cultural space. A visitor of a museum may wish to know which screens are free to watch, an employee of the cultural space would like to know the closest available printer etc. Apart from discovering devices, users often are interested to know the available services, e.g., which actions are available for the nearby ticket vendors. Finally, discovering users, such as friends or visitors of similar needs may be required in certain circumstances. Overall, discovering services, devices and users is a highly appreciated context-aware service provided by the proposed middleware which supports both on-demand and push-based triggering.
- *Authorize*. Authorization of restricted areas or actions is in high demand across context-aware environments. A visitor should not try to access restricted areas of a museum and upon entry the museum administration would like to be notified. Users with different profiles may or may not have the required privileges to perform certain actions, especially about publicly shared devices. Overall, the authorization of user actions and site accessibility in relation to user profiles is supported in the proposed middleware.
- *Recommend*. Recommender systems try to suggest points of interest (POI) or paths of POIs to visitors according to their preferences and other context such as the traffic of the area. Recommender systems are in essence context-aware applications that exploit context-aware middleware to acquire the necessary context and perform the recommendation process efficiently. A first layer of recommendation can be performed at middleware level, thus a minimal recommendation algorithm for POIs, paths of POIs and services has been included in the proposed middleware. In our previous work, cultural recommenders have exploited the extracted Persona of the visitor [13].

A data flow of the context reasoning process is illustrated in Fig. 5. There are three starting points that trigger the process. Either context is requested, new context is acquired, or a periodical execution is triggered. Different components of the reasoning module are responsible to tackle each triggering condition.

The temporal initiation is in essence a timer which triggers the rules checking every X minutes. The number X is specified by the middleware administrator and is related to the nature of the ecosystem and its sensitivity to extreme and quick changes. The identification of new context that enables context reasoning is performed at the context acquisition module. Not every new context triggers this, but for example the opening of a door initiates an exciting discussion on behalf of the middleware about who entered the room and what are their intentions. New context also initiates procedures to ensure quality of context and production of high-level context.

On the other hand, on-demand reasoning requires that a listener is executed and waits for incoming requests from the Context Provider Module. Upon such a request, the listener decides which type of service satisfies the requested context. Thus, authorization, discovery, recommendation or customization is selected.

The next step of the data flow is common to both real-time and on-demand reasoning. The selected service is associated with an equivalent reasoning technique. For example, quality of context will be tackled by the Uncertainty Module, while a request for activity recognition of the user will be tackled by the Machine Learning Module. The Inference Engine is called and the response is received. Many times, a service requires multiple calls from the inference engine, thus a cycle of inferences is initiated until the required contextual information is finally produced. The final output is disseminated to either the actuators or the context consumers through the Context Provider Module.

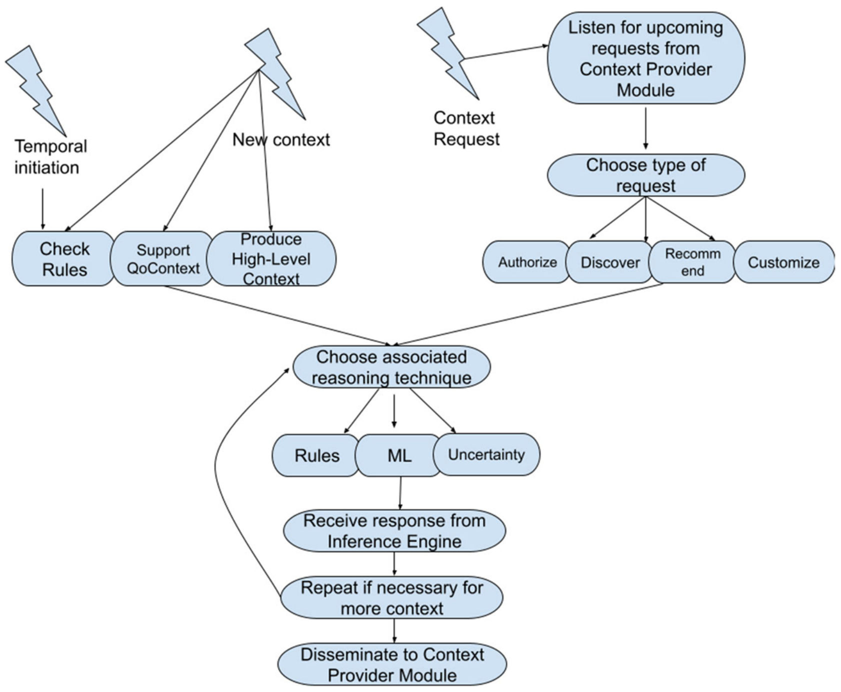

Temporal initiation
New context
Context Request
Listen for upcoming requests from Context Provider Module
Choose type of request
Check Rules
Support QoContext
Produce High-Level Context
Authorize
Discover
Recommend
Customize
Choose associated reasoning technique
Rules
ML
Uncertainty
Receive response from Inference Engine
Repeat if necessary for more context
Disseminate to Context Provider Module

Fig. 5 Context reasoning data flow

4 Experimental setup

4.1 Enhancing user interaction

The concept of smart cultural spaces "lies in the blending, not the separation, of the virtual and the real world", according to Falk and Dierking [42]. Enhancing the user interaction and measuring such enhancements is not straightforward. There is no quantitative methodology that can identify all the parameters, features and actions included in the explicit or subtle interaction between users and systems [43]. From a theoretical perspective, a visitor's experience can be seen in terms of *context, self-projection, embodiment, re-enactment, historicity* and *possibilities of being* [44]. Furthermore, it is even harder to quantify the gain or loss between different types and layers of interactivity, when complex interaction models are applied, such as in the IoT paradigm.

When limiting the scope of interactivity, appropriate methodologies for the measurement of the user interaction can be utilized. Othman et al. [45] have proposed the Museum Experience Scale which measures the engagement offered by digital guides to the visitors of museums. According to their research, there are four components that should be promoted: *engagement, meaningful experience, learning* and *emotional connection*. A similar approach has been adopted in another smart guide evaluation [43]. Another scheme of user experience enhancement through social participation includes scenarios that promote co-construction, generation and sharing [46] and co-creation [19].

When changing the focus from the smart cultural space to the smart interaction in generic ubiquitous environments, a few more indices are proposed in the literature. Carvalho et al. [47] propose the following measures: *adaptation correctness, availability, relevancy, context-awareness timing* and *courtesy*. The above indices were adapted and expanded from calm computing to Internet of Things environments [48], which is suitable for the ambient requirement of cultural spaces. Thus, such measurements that evaluate those indices are necessary in order to evaluate the user interaction provided by any smart environment scenario.

The proposed middleware implements a number of context reasoning services indicated in Fig. 4. Those services support different kinds of user interactivity scenarios, which can be categorized in the following five types: *customization, discovery, recommendation, real-time monitoring* and *quality of context*. Following the interactivity indices proposed by [47], correlations between services and indices can be identified.

- Customization is related to *adaptation correctness* and *courtesy*, since it allows the system to adapt to the user preferences, promoting courtesy by means of environmental and device customization that favors the visitor.
- Discovery is related to *availability*, presenting available devices, users or services to the visitor according to contextual parameters.
- Recommendation is related to *relevancy*, since it offers the identification of relevant paths or services to visitors.

- Real-time monitoring supports *context-aware timing* by timely adapting the devices to the user's needs.
- Finally, quality of context is an intrinsic service which supports all other services.

The experimental setup which will be described in the next subsections is designed and implemented in order to assess the enhancement of the user interaction through the context-aware functionality offered by the middleware. There are three basic questions that need to be answered:

- How accurate is the context-aware reasoning in identifying contextual information and acting upon it in favor of user interactivity in a smart cultural space?
- How do actions performed by the middleware or suggestions to the user enhance user interactivity when compared to non-contextual environments?
- How well does the proposed middleware perform in terms of delay and scaling, which may affect user interactivity?

The above questions will be explored in Section 6.

4.2 Description

The proposed middleware is designed to function in generic situations, where users interact with objects in predefined ways and machine-to-machine communication is also standardized. Nowadays, IoT installations can be found in various fields such as smart cities or smart industries. Smart cultural spaces have also lately emerged, promoting personalization of visitor interaction with tangible and intangible cultural heritage [21]. For the purposes of a case study of the proposed middleware, a smart cultural space is ideal since it encompasses most parts of an IoT ecosystem, in a semi-isolated environment, where users and objects are profiled and the supported activities are predefined. The dynamic nature of a cultural visit though, allows interesting scenarios that will evaluate the efficiency of the proposed middleware.

The cultural space of the case study includes both indoors and outdoors areas. It accommodates tangible artifacts and digital resources, such as screens and speakers. Some areas are restricted to personnel only, while some other areas have restricted access to specific sections. The cultural space is equipped with devices that control environmental parameters for indoor areas. All visitors are expected to register upon entry to the space, using a mobile device as part of the IoT ecosystem. Profiles are available for all visitors, past and current, all personnel and all active objects and devices located in the cultural space.

The visitors may visit the different areas of the cultural space in any order. They may examine tangible objects, interact with digital resources, engage in activities provided in the cultural space, use facilities or buy memorabilia from the shop. The cultural space provides a WiFi network that covers all areas, which is available to visitors, personnel and connected devices. Most of the context-aware services described in Section 3.3. Context Reasoning are supported by the middleware. Specifically, a visitor may request a list of devices or services available in the area or request for recommendations on artifacts or a visiting path. Access to restricted areas or sections is monitored, while authorization of actions is also supported. The middleware periodically checks the conditions of rules and performs equivalent actions upon validity, related to the visitors, personnel or artifacts of the cultural space. The middleware also analyzes new context to produce high-level context, identifying situations like the activity of a visitor. Finally, the mobile application and the installed devices are constantly customized in order to satisfy the combined profiles of the involved visitors.

4.3 Dataset and scenarios

The evaluation of an IoT installation requires a dataset which will include large amounts of sensed and profiled data, as well as other contextual data in order to cover many scenarios and situations. Testing the prototype in real-time scenarios offers higher degrees of unpredictability in contrast to simulated scenarios. On the other hand, real-time testing is time consuming, is more difficult to simulate extreme situations and is not suitable for scalability and performance evaluation. The current evaluation focuses on performance, utilizing a blend of real and artificial data in order to simulate experiments that capture the dynamics of a smart cultural space, while testing the efficiency of the prototype to tackle possible scenarios of varying importance.

The dataset used for the experiments consists of the following:

- profiled data for 550 cultural visitors and 30 personnel. This dataset was acquired by
- an ongoing project on cultural routes.
- profiled data on 350 cultural artifacts. This dataset was acquired by public repositories.
- profiled data on the areas, devices, sensors and other objects of the cultural space. This dataset was created manually.
- behavioral data for the users. This dataset was created artificially as follows: a number of visitor behaviors were initially defined along with their movements, paths and actions. Those initial behavioral patterns varied in order to capture as many types of visitors as possible. Parameters were defined for each initial pattern, such as time delay between stops. Randomizing

those parameters and combining different behavioral patterns, 1600 visits were created, each one associated with a user. Those artificial visits were timestamped in order to provide a flow of traffic in the cultural space during 2 weeks of about 100 visitors per day.

- rules and other contextual data. This dataset was created artificially, adding rules to profiles, following a similar method with behavioral data. From an initial set of rules, a parameterized combination was applied to users, artifacts, devices and other objects.
- sensed data for sensors. This dataset was created artificially as follows: periodical measurements of all sensors of the experiment were captured and stored. Artificial incidents were added in order to represent abnormal situations. Further editing such as the addition of false measurements or the removal of measurements was also performed. The final enhanced dataset was timestamped to match the behavioral data of visitors
- external data. This dataset was created artificially, including weather data which were timestamped and matched to the other timestamped datasets.

The creation of the above dataset involved the modelling of various scenarios which would capture as many situations as possible and evaluate the reaction and performance of the proposed middleware. The different scenarios tackled many aspects of a cultural visit, such as visiting paths, availability of services, activities, resources and devices, traffic at POIs, environmental incidents, recommendations and weather conditions. A non-exhaustive, representative list of scenarios implemented in the datasets is described below.

Scenario 1

Type: Customization
Description: Users visit the cultural space and the digital resources (mobile app, devices) are customized accordingly.
Scenario: A user enters an empty room of the museum.
Expected outcome: Based on the user's profile and the room's environmental parameters, the air conditioning is set to the optimal level. The volume of the mobile application is customized accordingly.
Variations: A user enters a room filled with other visitors as well, thus the combined preferences for environmental parameters are applied, while the volume of the user's mobile device is decreased to avoid disturbing the others.

Scenario 2

Type: Discovery
Description: Users request available nearby devices with digital resources.
Scenario: A user, located in a room of the cultural space, requests available interactive multimedia resources.
Expected outcome: Based on the user's profile and location, a list of available devices that support interactive activities is presented to the mobile app.
Variations: A user requests nearby available printers to print a personalized photo of the visit or an activity performed. A user requests available users who are interested in performing a collective activity.

Scenario 3

Type: Recommendation
Description: Users request recommendations for nearby artifacts.
Scenario: A user requests a recommended path for the artifacts of the museum, while the preferred duration of the visit has also been set.
Expected outcome: A recommended path based on the user's profile, the duration of the visit and the availability of resources, based on the current traffic is calculated and presented to the user's mobile app.
Variations: A user requests recommended activities.

Scenario 4

Type: Real-time monitoring
Description: Users move around the outdoors areas of the cultural space. One of the areas has restricted access to personnel only. The position of each user is constantly checked in order to identify prohibited user entries.
Scenario: A user with role: visitor is trying to enter a restricted area.
Expected outcome: A notification is sent to the user's mobile device and to the guard.
Variations: A user is performing an action, which is checked against his/her profile, e.g., trying to turn on/off a public screen.

Scenario 5

Type: Quality of Context
Description: Multiple sensors that measure the same parameter are fused in a single value.
Scenario: Room A has two sensors that measure the temperature, one in each exit. Each sensor transmits a value every 30 min.
Expected outcome: The raw sensory data are fused, according to sensor reliability and the calculated temperature is stored to the database.
Variations: A sensor stops transmitting measurements. A sensor skipped a couple of measurements the past hours.

The scenarios and multiple variations described above were coded into the dataset. The reaction of the middleware in each scenario was evaluated in the following three ways:

1. First, all context spanning the 2 weeks of the experiment was inserted into the database. The system's time was programmable and for each scenario it was set to match the timestamp of that scenario. This way was used in order to test and optimize specific scenarios.
2. Secondly, the middleware was continuously executed for a 2-week period, during which all coded scenarios were implemented and evaluated in real-time (with the same artificial data). This way was used in order to run a complete and continuous test of the system. In both ways, the exterior components connected to the middleware, such as smart devices and the mobile app were coded as software dummies that sent and received predefined data. Also, each scenario was tagged with an optimal context output which was compared to the actual middleware output, in order to calculate performance and accuracy metrics.
3. Finally, the middleware and supported infrastructure of the IoT ecosystem were implemented and evaluated in a simulated environment, with scenarios executed in real time. The performance and accuracy of the middleware were manually recorded by the participating users. This method was used to evaluate the connectivity and performance of the middleware under real conditions.

5 Validation

5.1 Infrastructure

For the purposes of the case study, a Wireless Sensor Network was implemented which included sensor nodes based on the Arduino product line. Arduino is a widely used open-source microcontroller development platform which offers flexible, low-cost and easy to use hardware components suitable for environmental monitoring applications [49, 50]. The sensor nodes consist of an Arduino MKR WiFi 1010 microcontroller and a sensor shield, depending on the needs. The MKR WiFi 1010 is based on the Arm Cortex-M0 32-bit SAMD21, a low-power MCU clocked at 32MHz. The board supports both Bluetooth Low Energy (BLE) and WiFi connectivity with NINA-W10, a low-power chipset operating in the 2.4GHz range. Secure communication is achieved with a Microchip ECC508 crypto chip. A Li-Po Single Cell, 3.7V battery can substitute the USB port for autonomous power supply.

On top of the MKR 1010 board, variable sensors are attached depending on the needs of each sensor node. The most common sensor used was an Arduino MKR Env Shield which allows the board to acquire environmental data collected by an array of sensors. The shield includes sensors that measure atmospheric pressure, temperature and humidity, ultraviolet UVA intensity, ultraviolet UVB intensity, UV Index (calculated) and light intensity (in LUX). The shield also supports local storage to a microSD card for projects that do not have a consistent network connection.

For the longevity of battery life and minimal maintenance costs, the Arduino MKR WiFi 1010 board has been programmed with low-power optimization coding, exploiting the *ArduinoLowPower* Library and the WiFi *LowPowerMode*. Use of routines such as *WiFi.lowPowerMode()* allows the WiFi NINA Module to reduce its power drain, bringing the overall power consumption to 30 mA. Yet, no functionality is lost since the module is still able to receive incoming data and broadcast the beacon signal, keeping the connection alive.

The server of the proposed middleware can run on any machine connected to the network. The computing requirements, mainly because of the execution of ML algorithms, are high enough to discard the use of typical low-power IoT solutions like Raspberry Pi, thus requiring the use of a PC acting as the server responsible for the data collection and reasoning. For the testing scenarios the role of the server was appointed to a Mac Mini equipped with an Intel i7 processor with 16GB RAM. The server was programmed in Python exploiting the scalability and interoperability of the language, as well as its wide support for ML projects.

5.2 Implementation

5.2.1 Context acquisition and dissemination

Most sensor nodes installed in the cultural space were connected through a WiFi network. Some devices, such as beacons, utilized BLE technology to communicate their signal to the server. The messaging protocol used in the middleware was Message Queuing Telemetry Transport (MQTT), a standardized publish/subscribe push protocol, released by IBM in 1999. It runs over (TCP/IP) offering ordered lossless connections. A typical MQTT architecture includes a client and a broker. MQTT is best suitable in constrained environments like embedded devices with limited processing capabilities and in unstable networks [51]. The MQTT broker Mosquitto was used in the current case study.

Apart from the sensory network, other devices such as the visitors' mobile devices, smart devices installed in the area and devices that presented digital resources to visitors communicated with the middleware. Again MQTT was the messaging protocol used, despite the non-constrained nature of those devices, in order to allow continuity among the smart objects of the ecosystem. A data flow of a

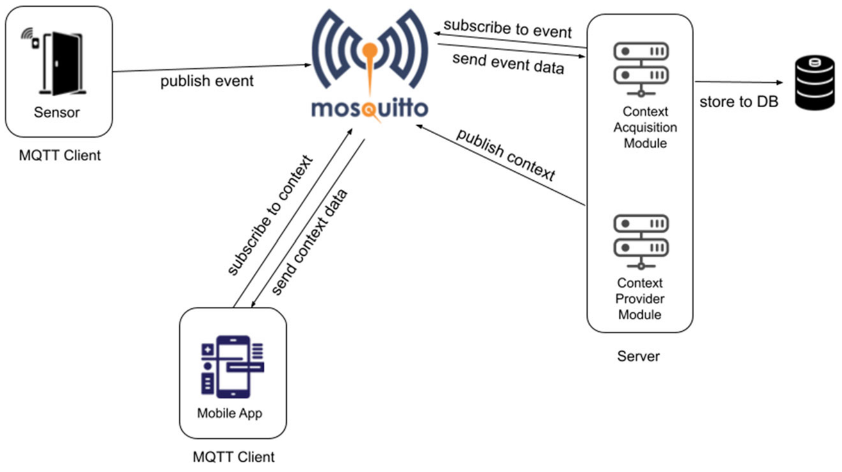


Fig. 6 MQTT context acquisition and dissemination data flow

typical context acquisition and dissemination procedure using MQTT in the proposed middleware is illustrated in Fig. 6.

5.2.2 Context modelling

The context model presented in Section 3.2 was implemented with MongoDB [1], a NoSQL database which is document oriented and supports high flexibility in data structures and dynamic querying. Furthermore it is very scalable, allowing large-scale, highly available and robust systems [52]. MongoDB has been exploited in many IoT installations such as storing context from sensors [53] or supporting context-aware recommender systems [54].

MongoDB organizes data in collections, similar to tables of relational databases. Each core class included in the context model and each subclass associated with the core classes are implemented as different collections. Figure 7 depicts an instance of a *location* object that represents one of the outdoors areas of the museum. The depicted location is geolocated using GPS measurements of the rectangle area, which is useful for positioning visitors inside that area.

Apart from storing fixed data, such as locations and things, the proposed schema supports modelling of roles and profiles, which indirectly enable rules such as authorization, accessibility and preferences. Figure 8 depicts an instance of a *role* object which is associated with a thingID. More specifically, the depicted role is “museum guard” which is associated with a specific user (*thingID*), for specific locations (an array of *locationClasses* that include all indoor and outdoors areas of the museum) and for specific timeframes (*time field*, indicating May to September, all days apart from Sunday and specific working hours). Thus, all privileges associated with this roleID (stored in other collections of the DB) are active as long as location and time conditions are met for the certain user.

Finally, one last instance of the database is shown in Fig. 9, which depicts an *activity* object. In this case, the activity is a sensor measurement and more specifically a GPS measurement associated with a *thingID*. The stored value is time stamped. Context data acquired directly from sensors are raw and have not yet been analyzed. During the context reasoning stage, the low-level context of Fig. 9 will be transformed to high-level context information about the positioning of thingID in relation to the museum areas. The user associated with the thingID (e.g., with a relationship “worn”) will also be attributed with the calculated positioning of the thingID.

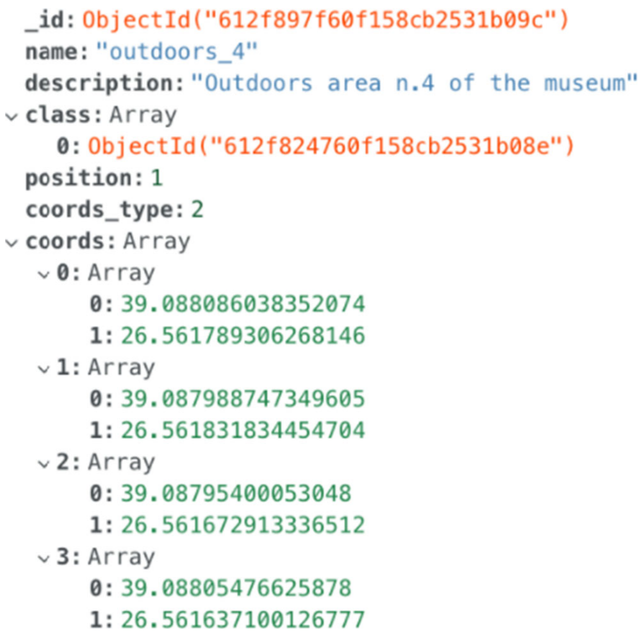
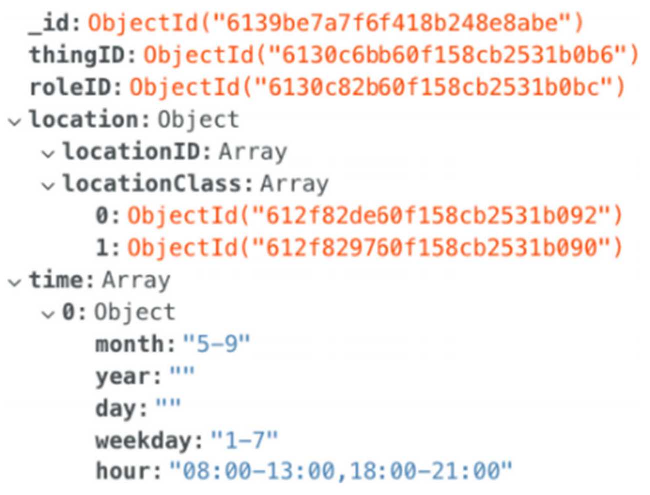

```
 _id: ObjectId("612f897f60f158cb2531b09c")
 name: "outdoors_4"
 description: "Outdoors area n.4 of the museum"
v class: Array
   0: ObjectId("612f824760f158cb2531b08e")
 position: 1
 coords_type: 2
v coords: Array
  v 0: Array
      0: 39.088086038352074
      1: 26.561789306268146
  v 1: Array
      0: 39.087988747349605
      1: 26.561831834454704
  v 2: Array
      0: 39.08795400053048
      1: 26.561672913336512
  v 3: Array
      0: 39.08805476625878
      1: 26.561637100126777
```

Fig. 7 Instance of a Location object in MongoDB

[1]https://www.mongodb.com/



```
  _id: ObjectId("6139be7a7f6f418b248e8abe")
  thingID: ObjectId("6130c6bb60f158cb2531b0b6")
  roleID: ObjectId("6130c82b60f158cb2531b0bc")
⌄ location: Object
  ⌄ locationID: Array
  ⌄ locationClass: Array
      0: ObjectId("612f82de60f158cb2531b092")
      1: ObjectId("612f829760f158cb2531b090")
⌄ time: Array
  ⌄ 0: Object
      month: "5-9"
      year: ""
      day: ""
      weekday: "1-7"
      hour: "08:00-13:00,18:00-21:00"
```

Fig. 8 Instance of a Role object in MongoDB

5.2.3 Context reasoning

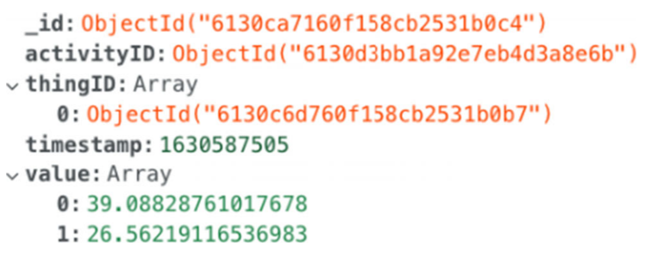
The context reasoning process is split into two subprocesses, as described in Section 3.3, real-time reasoning and on-demand reasoning. The basic difference is that during real-time reasoning the middleware is responsible to initiate context-aware procedures, such as rule checking, while during on-demand reasoning, the applications are responsible to request the required context.

The whole reasoning process is implemented with Python 3.7.4. Python[2] has received much attention the last decade due to its robust, dynamic code, which supports improved productivity and a vast community that offers open-source libraries and support. The implementation of the middleware included the use of many Python libraries such as *pymongo* for MongoDB support, *shapely* for geolocations, *numpy* for data manipulation, *paho-mqtt* for MQTT support, *socket* and *bluetooth* for network connectivity and *sklearn* for machine learning algorithms.

Real-time reasoning is implemented with a script that is executed every X seconds. The exact number of seconds is parametric and is calculated based on the dynamic nature of the ecosystem. In the current case study, due to the dynamic movement of visitors in the cultural space, the script is executed every 15 s, a time interval which, through experiments, achieved a balance between timely reaction and computational stress. Since real-time reasoning is computationally expensive, a dedicated localization module is exploited in order to undertake this stressful procedure.

Furthermore, on-demand reasoning is implemented with a script which listens for context requests from the Context Provider Module. As illustrated in Fig. 6, this module publishes context to the Mosquitto broker which then disseminates it to the interested context consumer. The on-demand reasoning module chooses the type of request and calls the inference engine accordingly, in order to produce the requested context service.

[2]https://www.python.org/



```
  _id: ObjectId("6130ca7160f158cb2531b0c4")
  activityID: ObjectId("6130d3bb1a92e7eb4d3a8e6b")
⌄ thingID: Array
    0: ObjectId("6130c6d760f158cb2531b0b7")
  timestamp: 1630587505
⌄ value: Array
    0: 39.08828761017678
    1: 26.56219116536983
```

Fig. 9 Instance of an Activity object in MongoDB

Both techniques, real-time reasoning (push) and on-demand reasoning (pull) are supported by the middleware, allowing the triggering of events either by the user or the system. Some rules may be executed based on both push and pull triggering, e.g., a video is reproduced on a screen in a room because a sufficient number of people entered the room or because a small group of visitors requested it. In general pull triggering is attributed with higher priority, allowing better control of the customization process to the visitor.

The inference engine, called by either the real-time or the on-demand reasoning process consists of three modules, each one exploiting a different reasoning technique. Each one is also implemented in Python. The selection of the appropriate technique is performed based on predefined conditions, related to the type of context service. Thus, if a context consumer requests the activity of the user, the inference engine executes the ML module, while if the request is for a rule to be checked, the Rules module is executed.

The context-aware services provided by the middleware are summarized in Table 1 and follow the services depicted in Fig. 4. Each service is implemented by a Python function, which is part of the inference engine. Part of the coding that implements the function that finds services in a location is shown in Fig. 10.

6 Results

The proposed middleware was evaluated using the scenarios and their variations described in the previous subsection. The behavior of the context-aware procedure was evaluated in respect to the following criteria: (a) accuracy, (b) performance and (c) interactivity enhancement.

6.1 Accuracy

Accuracy is defined as "...the degree of approximation to a certain expected value" and is applied to a large set of results

Table 1 Context-aware services and implementation details

Function	Description of service
Eval_rule	Evaluates the validity of a rule. Each rule has a set of profiles which represent the conditions and a set of actions that will be performed. A rule profile may have multiple conditions, such as temporal or spatial restrictions, environmental boundaries, location of things and triggering activities. Each rule decision is accompanied by a confidence factor, depending on the oldness of measurements and other parameters.
Check_profile	Checks the profile of a thing and executes actions that are associated. Each profile may have multiple conditions, such as temporal or spatial restrictions and proximity to other things. Each profile also has an aggressiveness parameter, which represents an importance factor and is compared to the confidence of the decision. Thus, a less aggressive profile will need higher confidence from the inference process to be validated and the associated actions to be performed.
Check_action_auth	Checks whether a specific action about to be initiated by a user is valid. Actions may include entering an area or manipulating things. Each action is associated with a notification action in case of prohibition
Find_services find_things find_users	Discovers services, things and users based on the caller's location. Services are activities related to things of the location, such as turning on a display, playing a sound and raising the volume of speakers. The discovery is based on privileges and preferences, which is again checked by the profiles of both caller and targets. For example, a visitor may be privileged to manipulate all smart screens of the area, but a specific screen may be off limits.
Validate_context	A complex function which performs quality of context control. It consists of a few processes that (a) fill missing context gaps based on historical data (exploiting a Hidden Markov model), (b) evaluate the quality of incoming data and remove erroneous values and (c) combine different sources of the same contextual topic based on sensor reliability and oldness of measurement (the Dempster-Shafer algorithm is exploited).
Calculate_location	Calculates the location of a thing, based on sensed data, such as GPS measurements or BLE proximity. It also calculates recursively the location for any user that holds or wears this thing. In case of movable things, a confidence factor is also calculated, which is 1.0 (max) for fixed things, e.g., an installed device.
Identify_activity	A complex function which identifies the activity of the user, based on various parameters, such as latest measurements of the associated GPS or other location sensor, the interaction of the user with other devices and explicit user input. A Random Forest supervised learning algorithm provided by the sklearn library was utilized for the activity recognition [55].
Identify_abnormal	Identifies abnormal situations based on incoming sensed data. For example, given a series of temperature measurements of a museum room the past few days, a possibly abnormal measurement is tagged and the appropriate users are notified. An Isolation Forest unsupervised algorithm provided by the sklearn was utilized in the outliers detection process [56].

[57]. It is commonly used in machine learning to measure the ability of proposed models to solve the problem in hand with sufficient efficiency. Furthermore, accuracy has been identified as a major aspect of evaluation for context-aware decision support systems. It is crucial to assess the frequency of users encountering inaccurate matters, such as imprecise results, inconsistencies and differences between actual and expected results [58].

The proposed middleware performs a large number of context inferences, from identifying the location of users, to customizing devices or acting upon the validation of a rule. In order to measure the accuracy of those inferences, i.e., to check whether the inference outputs were correct, each inference instance needed to be tagged with the correct output. The dataset described in Section 5.1 includes not only the raw contextual data, but also expected context information. Table 2 shows a few representative records of contextual data and the expected inference output.

The tagged dataset utilized in the evaluation of each scenario, allowed the measurement of accuracy, in terms of the percentage of correct outputs of the inference process. In order to measure accuracy, the use of predefined scenarios was preferred against the use of spontaneous scenarios, since the latter would require real-time evaluation of each inference output by the user, an impossible task. Besides, the predefined scenarios were created with artificial data and were flavored with real-world features such as inaccuracy, inconsistency and variability.

The measured accuracy was categorized in several distinct functions of the inference engine, closely related to the context reasoning services of Fig. 4. Those categories are as follows: identification of user location for indoor and outdoors areas, fusion of multiple sensed data, filling of missing sensed values, identification of validated rules, discovery of requested devices, discovery of requested services, discovery of requested users, recommendation of

Fig. 10 Python code of the find_services function

```python
def find_services(self, locationID, passive=False):
    services = []
    location_classes = self.getLocationClasses(locationID)
    things = self.findThingsInLocation(locationID)
    for thing in things:
        activities = []
        #for each thing find the activities
        if passive:
            activities_thing = self.getEntities("activity_thing",
                                        "thingID", thing["thingID"])
        else:
            activities_thing = self.getEntities("activity_thing",
                "thingID", thing["thingID"], "type", TYPE_ACTIVITY_ACTIVE)
        #for each activity check for profile
        for activity_thing in activities_thing:
            check = True
            profile_activities = self.getEntities("profile_activity",
                                    "activityID", activity_thing["_id"])
            for profile_activity in profile_activities:
                #check if profile is valid for time and location
                if not self.checkLocationRestriction(profile_activity["location"],
                                          locationID, location_classes) or \
                   not resolveTime(profile_activity["time"]):
                    check=False
                    break
            if check:
                activities.append(activity_thing["_id"])
        services.append({"thingID":thing["thingID"], "activityID":activities})
    return services
```

POIs, recommendation of services, user activity recognition, identification of required customization. The measured accuracy in each category is presented in Table 3.

The measured accuracy depicted in Table 3 indicates that the proposed middleware shows high accuracy in most categories of context-aware services. Such tasks that are highly affected by more static data, i.e., those tasks that have low correlation to user behavior and movement, show accuracy scores of more than 95%. Thus, context-aware services such as identifying abnormal environmental situations, customizing environmental devices, discovering services and devices have a very high success rate and the behavior of the middleware can be considered optimal.

Those tasks that have higher degrees of human behavior, show accuracy of among 89–96%, with most of those scores being above 90%. This is indicative of the much more dynamic characteristic of scenarios when the user's movement is involved. The unpredictability and irregularity of user behavior, even if artificial, causes seldom erroneous inferences on behalf of the context-aware reasoning process, in tasks such as recommendation of services for the user, customization of their mobile device and filling missing values about user actions.

The indoor localization process shows better accuracy than the outdoor localization process, since indoor positioning depends on more precise technologies, such as BLE, while the GPS used in outdoors areas proves less precise in positioning the user correctly. The identification of user activities, tackled by machine learning algorithms shows a sufficient 89% accuracy, which can be improved with better training of the model, a task that lies out of focus of this research work.

Overall, the results on the measured accuracy are very positive, indicating that the middleware behaves as expected

Table 2 Part of Dataset tagged with expected inference output

Raw data	Expected output	Description
(long, lat) of userID	set (location of thingID) to userID location	The location of a user is transformed from gps coordinates to proximity to a POI.
(long, lat) of userID	execute actionID	The movement of a user at the current location causes the validation of a rule and the execution of an action (in this case, a violation of non-access entry).
(value) of sensorID	set (value) to environmental parameter	The acquisition of sensory data (e.g., temperature) initiates a fusion with stored data of correlated sensors, with the output value stored as the environmental parameter (Quality of Context).
(selection) of userID	return (list of devices)	An incoming request for selected nearby devices with multimedia content from the user causes the production of a list of such devices.

Table 3 Accuracy of inference output per category

Category	Subcategory	Accuracy
Identification of user location	indoors	98%
Identification of user location	outdoors	96%
Fusion of multiple sensed data	–	98%
Filling of missing values	environmental	99%
Filling of missing values	user behavior	89%
Identification of validated rules	authorized entries	96%
Identification of validated rules	authorized actions	100%
Identification of validated rules	abnormal environmental situations	100%
Discovery	services	98%
Discovery	devices	99%
Discovery	users	97%
Recommendation	POIs	96%
Recommendation	services	93%

in almost all tasks. Besides, critical services, such as the identification of validated rules and the customization, which both lead to automated actions, showed very high accuracy, minimizing the risk of erroneous actions on behalf of the system. Finally, categories such as recommendations and discovery also showed high accuracy, which leads to a better user experience.

6.2 Performance

Apart from providing an accurate inference engine, a context-aware middleware also needs to achieve performance in terms of two metrics: reaction time and scalability. The first measures how quickly the middleware identifies and acts upon events that need to be tackled, while the second measures the ability of the middleware to scale well with the increase of IoT nodes and rules. In order to test the performance for both metrics, the middleware calculated and stored the delay in each scenario and each inference output. The results of the experiment will be provided in the next paragraphs.

For the calculation of the reaction time of on-demand reasoning, the Context Provider Module stored the timestamp of each context request and the timestamp of the equivalent inference output. On the other hand, the calculation of the reaction time of real-time reasoning was based on the timestamp of the inference output and the timestamp of the acquisition of context data related to the event. Thus reaction time was measured as the delay between cause (acquired context or request) and effect (actuation). Table 4 summarizes the average reaction time per service category.

The results indicate that the middleware is almost instant when dealing with both on-demand context requests and real-time reasoning. Those categories that do not suffer from performance loss, such as quality of context procedures are omitted from the performance tests. Customization is the only category that shows increased reaction times, which is mainly caused because customization is usually triggered by profiles and rules checking, a task that was performed in a 15-s cycle during the experiments.

For the evaluation of scalability performance of the middleware, dataset entries were replicated multiple times in order to include larger numbers of IoT nodes, relationships between them and rules and profiles attached to them. Afterwards, similar scenarios as before were executed, this time requiring more calculations and inferences performed by the middleware. The reaction time was measured for all categories and the results are depicted in Table 5.

The reaction time, which is indicative of how quickly the middleware can respond to events that require context-aware computing, follows a linear correlation to the number of things (and rules). Figure 11 illustrates this correlation. As the number of things and rules increases, the average reaction time, which is highly related to the database calls, also increases. This increase though, following a linear correlation, was expected and proves the scalable feature of the proposed middleware. The linear correlation may become an issue with large-scale IoT installations such as those that extend to a city, but is capable of dealing with

Table 4 Average reaction time per context-aware service category

Type of reasoning	Category	Reaction time (s)
Real-Time	Environmental rules	0.09
Real-Time	User-centric rules	0.35
On-demand	Authorization	0.10
On-demand	Discovery	0.18
On-demand	Recommendation	0.23
Both	Customization	2.45

Table 5 Average reaction time in relation to number of things and rules

Number of things	Number of rules	Reaction time (s)
1200	2000	0.28
2500	4000	0.35
5000	9000	0.67
10000	17000	1.34
20000	35000	2.57
50000	90000	6.01

constrained areas such as a cultural space. It proves that with the increase of IoT nodes, the middleware's delay is related only to the addition of new nodes, rather than the complexity of the system due to larger inference cycles.

6.3 Interactivity enhancement

The proposed middleware was designed in order to optimize the cultural user experience. Typically, this includes the enhancement of interactivity offered by the various devices installed in the cultural space or carried by the users. Measuring this enhancement is not an easy task, especially in cases of artificial scenarios. Still, an approximate measurement of the interactivity enhancement can be achieved by calculating the number of context-aware services offered to a user during his/her cultural visit. Table 6 summarizes the average number of services offered to the users per category, during a unique visit to the cultural space.

The findings of Table 6 are based on artificial data, which means that the number of context requests by visitors is manually entered, thus reducing the usefulness of the calculated figures of the table. It should be noted that the figures depict the actual cases in which a recommendation or discovery was approved by the visitor and not every recommendation by the system. On the other hand, the customization services were mainly automated and triggered by non-user events, which makes the related figures more useful. Overall, Table 6 is indicative of

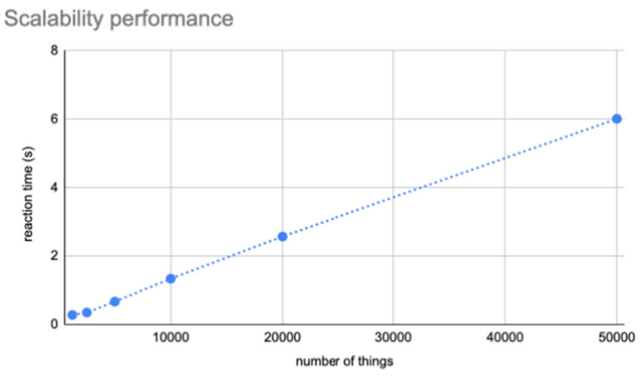


Fig. 11 Correlation between number of things and reaction time

Table 6 Average occurrence of services per user visit

Category	Average occurrence per visit
Authorization	2.6
Discovery of services/devices/users	5.6
Recommendation of POIs/services/paths	7.1
Customization of mobile device	5.8
Customization of environmental devices	1.2
Customization of cultural devices	3.4

the amount of context-aware services that the proposed middleware can offer in a cultural visit. It actually depicts what interactivity would miss if context-awareness was absent from the ecosystem. The user would not experience any customization of the smart environment while the recommendations and discovery services would be either absent or static for each visitor.

The disadvantage of the artificial dataset concerning the measurement of impact to the user experience is known to the authors, which is planned to be addressed by non artificial experiments. Nevertheless, the proposed middleware boasts that, depending on the user's needs, it can offer a wide range of context-aware services, enhancing the interactivity of the user with the smart cultural space.

7 Discussion

The main purpose of the proposed middleware is the enhancement of user interaction, with specific interest in cultural spaces, which is a dynamic environment where people interact in numerous ways and in a time and space constrained way. Furthermore, context-awareness allows a less intrusive way of the system to customize the dynamic environment in favor of a more natural user interaction with the cultural space. The results of the previous section indicate that the designed middleware is efficient in two ways (a) in accurately identifying a dynamic and multi-factorial context and (b) in achieving appropriate performance measured by reaction time even when scaled in large numbers of sensors. This section will discuss how the middleware showing such efficiency can lead to better user interaction. Table 6 is indicative of the number of context-aware occurrences that are directly involved in context-aware procedures of the middleware. Each category will be analyzed in the following paragraphs.

Authorization, as a process that controls the user profiles and their privileges to access areas, things and services, highly benefits from IoT installations. In cultural spaces, there are various types of users with appropriate profiles, such as visitors, guides, curators and guards.

Controlling the manipulation of smart devices or the access of restricted areas without context-aware computing, would require the excessive use of keys and passwords. The proposed middleware eliminates the need of such manual manipulations and offers instead automated control and provision of privileges on the spot. Thus, a lock is automatically opened when a guard tries to open the door of a restricted area, while a guide may have the right to control the environmental devices of a room, where his/her audience is located currently. Authorization has occurred 2.6 times per visit (average number independently of type of visitor) and while it may seem that this category mainly interests the institution's personnel, many occurrences are related to simple visitors. A non-exhaustive list of such occurrences considered in the proposed middleware include the following: the manipulation of a smart screen which is authorized only when there are no other unrelated visitors in the proximity, the manipulation of the sound of an exhibit when there is no conflict of interest with other visitors, the authorization of use of accessible toilets and the authorization of entry in rooms that offer activities restricted to adults.

Discovering services or devices offered by a smart environment is an exciting and rewarding experience, available through context-ware procedures. Visitors exploring a new place would like to experience as many moments as possible, according to their own preferences. Typically, such moments are related to services offered, e.g., a 3D spectacle, a simulation, an experiential activity, or related to available devices. Discovering such experience initiators is not always straightforward, especially when the user is overwhelmed or intimidated inside a cultural space. Context-awareness allows the discovery to be easy, automated and adapted to each user without effort. The proposed middleware has included the ability to search for nearby devices, or available services which are tailored to the user profile. Thus, the user interacts with the environment effortlessly and can identify a nearby screen, an interesting activity or the closest restroom. Furthermore, an even more interesting service is the discovery of users who may share similar interests and have expressed similar desires for connectivity. More specifically, the proposed middleware includes the ability to discover people who wish to participate in the same activity or follow the same tour. Overall, the discovery of services/devices/users is addressed efficiently by the middleware and has been measured to occur 5.6 times per visit.

Similar to discovery, recommendation is enhancing a user experience by providing a list of POIs, services or paths that are best suited to the user, according to their preferences and past behavior. Recommender systems are highly affected by context-aware feedback, since they can be adapted to the current user's needs. Although the inclusion of a sophisticated recommender system is out of scope of the current work, a minimal recommender system was included and exploited whenever the user was in need of such recommendations. The proposed middleware includes an option of recommended visiting paths and recommended services, a service which in contrast to the discovery service is not isolated in the cultural visit but views each visit as interconnected stream of activities. Thus, the user interaction with the smart cultural space is enhanced in the proposed installation by recommending paths which can be followed by the visitor and include POIs, activities or smart devices that augment the cultural visit, in a rate of 7.1 per visit.

Customization is considered the first level of context-aware services offered by IoT environments. Users interact with a number of digital devices, including the mobile application provided by the cultural space, smart devices offering digital experiences or environmental devices installed in the areas of the institution. In contrast to other services such as discovery or authorization, customization is passive since ideally the customization process is intransparent and not intrusive to the user. The proposed middleware includes various customization actions, such as the following: adapting the volume of the mobile device according to nearby people, customizing the user interface according to user behavior and past options, adapting the environmental devices that control room temperature according to the user preferences, adapting the volume/illuminance of smart screens according to the proximate users, customizing the content shown on a screen according to user profile (such as age, nationality, interests). The automated customization described above has been measured to occur 10.4 times per visit, accumulated for all types of customization.

The proposed middleware is designed to enhance user interaction in cultural spaces. Furthermore, the exploitation of context-awareness has enhanced the requirement for ambient incorporation of digital media in cultural spaces, a necessary feature that satisfies the post-digital museum. The previous discussion identified the ways that such an enhancement is delivered. The process includes the accurate identification of context by the middleware and the efficient manipulation of the context by cultural applications and devices. The former has been implemented in this research work, while the latter lies in the responsibility of cultural space designers. With the use of demo applications and devices, this research has shown that user interaction is enhanced when accompanied by a context-aware framework, such as the one provided by the proposed middleware.

8 Conclusion

In this research work, a context-aware middleware system that supports smart cultural spaces was presented, while novel context-aware modelling and reasoning techniques were integrated. The context reasoning procedure included real-time and on-demand reasoning, addressing context requests originating from users or the system itself. A hybrid reasoning schema was proposed, applying each individual technique to solve those problems that it is more suitable for.

The proposed middleware system provided context-aware services such as support of quality of context, production of high-level context, rules checking, customization, resources discovery, authorization and recommendation. The evaluation of the proposed middleware was based on a novel methodology which exploited artificial data that were combined with manual data which simulated circumstances that needed attention. The evaluation was performed by executing scenarios that ran in real time, but used artificial data stored in the database and time stamped accordingly.

The proposed middleware was assessed with respect to accuracy, performance and interactivity enhancement. The findings show that the inference engine is highly accurate when addressing more static problems, while the accuracy almost always stays above 90% for all services. The most critical services show even better accuracy scores, minimizing the risk of inconsistent system behavior. The evaluation also indicated high performance of the middleware in terms of reaction time, while also showing linear correlation to the number of things, a finding which proved the scalability of the system. Finally, the enhancement of interactivity and the gain in user experience was measured as very satisfying, but since experiments were based on artificial data, those findings are less useful. Future work will include the installation of the proposed middleware in a real-world cultural space and its evaluation by users in terms of user experience and enhanced interactivity.

Acknowledgements This research has been conducted as part of the ongoing project CAnTi (MIS—5056234) of the Call RESEARCH – CREATE – INNOVATE of the Operation Program Competitiveness, Entreneurship and Innovation, Greece.

Funding Open access funding provided by HEAL-Link Greece.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Rahman MA, Asyhari AT (2019) The emergence of Internet of Things (IoT): Connecting anything, anywhere. Multidisciplinary Digital Publishing Institute
2. Sundmaeker H, Guillemin P, Friess P, Woelfflé S et al (2010) Vision and challenges for realising the internet of things. Cluster of European research projects on the internet of things. European Commision 3(3):34–36
3. Atzori L, Iera A, Morabito G (2010) The internet of things: a survey. Comput Netw 54(15):2787–2805
4. Yano K, Akitomi T, Ara K, Watanabe J, Tsuji S, Sato N, Hayakawa M, Moriwaki N (2015) Profiting from iot: The key is very-large-scale happiness integration. In: 2015 symposium on VLSI technology (VLSI Technology), IEEE, pp 24–27
5. Perera C, Zaslavsky A, Christen P, Georgakopoulos D (2013) Context aware computing for the internet of things: a survey. IEEE Commun Surv Tutor 16(1):414–454
6. Abowd GD, Dey AK, Brown PJ, Davies N, Smith M, Steggles P (1999) Towards a better understanding of context and context-awareness. In: International symposium on handheld and ubiquitous computing, Springer, pp 304–307
7. Razzaque MA, Milojevic-Jevric M, Palade A, Clarke S (2015) Middleware for internet of things: a survey. IEEE Internet Things J 3(1):70–95
8. Paridel K, Bainomugisha E, Vanrompay Y, Berbers Y, De Meuter W (2010) Middleware for the internet of things, design goals and challenges. Electronic Communications of the EASST 28
9. Bettini C, Brdiczka O, Henricksen K, Indulska J, Nicklas D, Ranganathan A, Riboni D (2010) A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing 6(2):161–180
10. Pradeep P, Krishnamoorthy S (2019) The mom of context-aware systems: a survey. Comput Commun 137:44–69
11. Li X, Martínez J-F, Rubio G (2017) Towards a hybrid approach to context reasoning for underwater robots. Appl Sci 7(2):183
12. Fidas CA, Avouris NM (2015) Personalization of mobile applications in cultural heritage environments. In: 2015 6th international conference on information, intelligence, systems and applications (IISA), IEEE, pp 1–6
13. Konstantakis M, Aliprantis J, Michalakis K, Caridakis G (2018) Recommending user experiences based on extracted cultural personas for mobile applications-repeat methodology. In: MobileCH@ mobile HCI
14. Michalakis K, Christodoulou Y, Caridakis G, Voutos Y, Mylonas P (2021) A context-aware middleware for context modeling and reasoning: a case-study in smart cultural spaces. Appl Sci 11(13):5770

15. Packer J, Ballantyne R (2016) Conceptualizing the visitor experience: a review of literature and development of a multifaceted model. Visitor Studies 19(2):128–143
16. Konstantakis M, Michalakis K, Aliprantis J, Kalatha E, Caridakis G (2017) Formalising and evaluating cultural user experience. In: 2017 12th international workshop on semantic and social media adaptation and personalization (SMAP), IEEE, pp 90–94
17. Parry R (2013) The end of the beginning: Normativity in the postdigital museum. Museum Worlds 1(1):24–39
18. Mason M (2020) The elements of visitor experience in post-digital museum design. Participatory Design: Principles and Practices 14(1):1–14
19. Holdgaard N, Klastrup L (2014) Between control and creativity: challenging co-creation and social media use in a museum context. Digital Creativity 25(3):190–202
20. Michalakis K, Caridakis G (2022) Context awareness in cultural heritage applications: a survey. ACM J Comput Cult Heritage (JOCCH) 15(2):1–31
21. Not E, Petrelli D (2018) Blending customisation, context-awareness and adaptivity for personalised tangible interaction in cultural heritage. Int J Human-Comput Stud 114:3–19
22. Falk JH (2016) Identity and the museum visitor experience. Routledge
23. Falk JH, Dierking LD (2016) The museum experience revisited. Routledge
24. Bitgood S (2016) Attention and value: Keys to understanding museum visitors. Routledge
25. Dim E, Kuflik T (2014) Automatic detection of social behavior of museum visitor pairs. ACM Trans Interact Intell Syst (TiiS) 4(4):1–30
26. Chen H, Finin T, Joshi A, Kagal L, Perich F, Chakraborty D (2004) Intelligent agents meet the semantic web in smart spaces. IEEE Internet comput 8(6):69–79
27. Badii A, Crouch M, Lallah C (2010) A context-awareness framework for intelligent networked embedded systems. In: 2010 Third international conference on advances in human-oriented and personalized mechanisms, technologies and services. IEEE, pp 105–110
28. Celdrán AH, Clemente FJG, Pérez MG, Pérez GM (2014) Secoman: a semantic-aware policy framework for developing privacy-preserving and context-aware smart applications. IEEE Syst J 10(3):1111–1124
29. ElKady M, Elkorany A, Allam A (2020) Acaiot: a framework for adaptable context-aware iot applications. Int J Intell Eng 13:271–283
30. Chianese A, Marulli F, Moscato V, Piccialli F (2013) Smartweet: A location-based smart application for exhibits and museums. In: 2013 international conference on signal-image technology & internet-based systems, IEEE, pp 408–415
31. Amato F, Chianese A, Mazzeo A, Moscato V, Picariello A, Piccialli F (2013) The talking museum project. Procedia Comput Sci 21:114–121
32. Piccialli F, Chianese A (2017) The internet of things supporting context-aware computing: a cultural heritage case study. Mob Netw Appl 22(2):332–343
33. Dossis M, Kazanidis I, Valsamidis S, Kokkonis G, Kontogiannis S (2018) Proposed open source framework for interactive iot smart museums. In: Proceedings of the 22nd pan-hellenic conference on informatics, pp 294–299
34. Amato F, Moscato V, Picariello A, Colace F, Santo MD, Schreiber FA, Tanca L (2017) Big data meets digital cultural heritage: Design and implementation of scrabs, a smart context-aware browsing assistant for cultural environments. J Comput Cult Herit (JOCCH) 10(1):1–23
35. Alletto S, Cucchiara R, Del Fiore G, Mainetti L, Mighali V, Patrono L, Serra G (2015) An indoor location-aware system for an iot-based smart museum. IEEE Internet Things J 3(2):244–253
36. Mighali V, Del Fiore G, Patrono L, Mainetti L, Alletto S, Serra G, Cucchiara R (2015) Innovative iot-aware services for a smart museum. In: Proceedings of the 24th international conference on World Wide Web, pp 547–550
37. Bartolini I, Moscato V, Pensa RG, Penta A, Picariello A, Sansone C, Sapino ML (2016) Recommending multimedia visiting paths in cultural heritage applications. Multimed Tools Appl 75(7):3813–3842
38. Not E, Petrelli D (2019) Empowering cultural heritage professionals with tools for authoring and deploying personalised visitor experiences. User Model User-Adap Inter 29(1):67–120
39. Henricksen K (2003) A Framework for Context-aware Pervasive Computing Applications. University of Queensland Queensland
40. Hariri RH, Fredericks EM, Bowers KM (2019) Uncertainty in big data analytics: survey, opportunities, and challenges. J Big Data 6(1):1–16
41. Buchholz T, Küpper A, Schiffers M (2003) Quality of context: What it is and why we need it. In: Workshop of the HP openview university association
42. Falk JH, Dierking LD (2018) Learning from museums. Rowman & littlefield
43. Petrie H, Othman MK, Power C (2017) Smartphone guide technology in cultural spaces: Measuring visitor experience with an iphone multimedia guide in shakespeare's church. Int J Hum-Comput Interact 33(12):973–983
44. Pallud J, Monod E (2010) User experience of museum technologies: the phenomenological scales. Eur J Inf Syst 19(5):562–580
45. Othman MK, Petrie H, Power C (2011) Engaging visitors in museums with technology: scales for the measurement of visitor and multimedia guide experience. In: IFIP conference on human-computer interaction, Springer, pp 92–99
46. Díaz P, Bellucci A, Yuan C-W, Aedo I (2018) Augmented experiences in cultural spaces through social participation. J Comput Cult Heritage (JOCCH) 11(4):1–18
47. Carvalho RM, Andrade RM, Oliveira KM (2015) Using the gqm method to evaluate calmness in ubiquitous applications. In: International conference on distributed, ambient, and pervasive interactions. Springer, pp 13–24
48. Andrade RM, Carvalho RM, de Araújo IL, Oliveira KM, Maia ME (2017) What changes from ubiquitous computing to internet of things in interaction evaluation? In: International conference on distributed, ambient, and pervasive interactions. Springer, pp 3–21
49. Ferdoush S, Li X (2014) Wireless sensor network system design using raspberry pi and arduino for environmental monitoring applications. Procedia Comput Sci 34:103–110
50. Zafar S, Miraj G, Baloch R, Murtaza D, Arshad K (2018) An iot based real-time environmental monitoring system using arduino and cloud service. Eng Technol Appl Sci Res 8(4):3238–3242
51. Soni D, Makwana A (2017) A survey on mqtt: a protocol of internet of things (iot). In: International conference on telecommunication, power analysis and computing techniques (ICTPACT-2017), vol 20
52. Medvedev A, Zaslavsky A, Indrawan-Santiago M, Haghighi PD, Hassani A (2016) Storing and indexing iot context for smart city applications. In: Internet of things, smart spaces, and next generation networks and systems. Springer, pp 115–128
53. Mehmood NQ, Culmone R, Mostarda L (2017) Modeling temporal aspects of sensor data for mongodb nosql database. J Big Data 4(1):1–35

54. Khan A, Ahmad A, Rahman AU, Alkhalil A (2020) A mobile cloud framework for context-aware and portable recommender system for smart markets. In: Smart infrastructure and applications. Springer, pp 283–309
55. Botros M, Heskes T, de Vries IA (2017) Supervised learning in human activity recognition based on multimodal body sensing. PhD thesis, Bachelor's Thesis, Radboud University, Nijmegen, The Netherlands
56. Liu FT, Ting KM, Zhou Z-H (2008) Isolation forest. In: 2008 Eighth Ieee international conference on data mining, IEEE, pp 413–422
57. Hofer M, Strauß G, Koulechov K, Dietz A (2005) Definition of accuracy and precision—evaluating cas-systems. In: International congress series, vol 1281, Elsevier, pp 548–552
58. Ayed EB, Ayed MB, Kolski C, Ezzedine H, Gargouri F (2015) Context aware criteria for the evaluation of mobile decision support systems. In: 2015 IEEE/ACIS 14th international conference on computer and information science (ICIS), IEEE, pp 1–6