A Survey on Context Aware Computing in Digital Ecosystems[1]

AVERIAN, Alexandru
"Politehnica" University of Bucharest,
Bucharest, Romania
aaverian@gmail.com

Abstract

Ecosystems are considered complex adaptive systems formed from diverse elements which interact at a local level, adapt to a medium and evolve through a selection process. The most interesting characteristics of an ecosystem are adaptability to a medium and its tendency to reach a dynamic balance. Adapting to a medium in an ecosystem assumes the existence of context-aware applications that interact to achieve a general global goal. A context-aware application has the ability to adapt in an autonomous way to the actual context, which is permanently changing, with the purpose of giving an answer and an optimal experience for its user. In this article we consider digital ecosystems as open systems of applications with a degree of context-awareness and we analyze the most relevant models of context from the perspective of digital ecosystems. This discussion is followed by a comparison of current context modelling and reasoning techniques.

Keywords: *context, context-awareness, digital ecosystems.*

ACM/AMS Classification: 94A15

1. Introduction

Applications that are currently being developed evolve in the direction where explicit user intervention tends to diminish, data entered is steadily replaced by data measured by a series of sensors, computers or devices distributed around application entities. In other words, the context in which the entity of interest, the physical and virtual environment in which the applications are running, and the context of the user who exploits the application have an increasingly important role in the construction and operation of new types of context-aware applications. The notion of context was introduced by Weiser in 1991 in the pervasive computing domain [1], which was the start in studying context aware systems. Formal research in context-awareness started

[1]Citation: Averian A., *A Survey on Context Aware Computing in Digital Ecosystems*, "An. Univ. Spiru Haret, Ser. Mat.-Inform.", 13(1), pp. 21-40, 2017.

with [2]. In the realm of artificial intelligence, the context appears as a set of logical constructs that facilitate deductions, more precisely defined as "everything that affects calculus less explicit input and output data," so the context of a system may change depending on what we consider implicit or explicit.

2. *Context definition*

Although we intuitively understand the notion of context, it is difficult to define, and there is no formal universally accepted definition, as it results from a recent review [3] which analyzes this concept and its modeling in literature. The most used definition is given by Dey [4] and is valid for context-aware applications that run in an digital ecosystem: *"the context consists of any information that can be used to characterize the situation in which an entity is located. An entity may be a person, place, or object that is considered relevant in user-application interaction, the context may include both the user and the application"*. Schilit defines three context categories: physical context, user context, digital or computational context [2]. Similarly, Chen and Kotz in [5] classify context in four major categories: computing, user, physical and time context. Context is defined as: *"the set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the user"*. Bazire and Brezillon analyze in [6] a list of over 150 definitions of context available in various domains trying to find a decontextualized context characterization (outside of any context). The above definition of Dey is on position 36. In [6]'s conclusion, the context acts as a "set of constraints that influenced the behavior of a system involved in a given process." In [7], context is *a constraint in solving a problem without explicitly intervening in solving it.* In [8], context and context-awareness is refined in the scope of the digital health ecosystems: *"context is a collection of information that entities in health systems (usually stakeholders, but also software applications) can use to characterize the situation of another entity (usually a person, but also a software application) in its environment and to reshape their interpretation of the state or behavior of this last entity".*

2.1. **The nature of context information**

Background information generally comes from three sources: users, sensors, or is deduced from other context data. Information from users may be static (does not change, has a high degree of accuracy), or dynamic, related to the user's profile and activities, is given by the user or some of the applications he uses (it is rarely changed, missing or obsolete). Information from sensors often changes and may be outdated or erroneous if it is not retrieved. The information deduced from other sources depends on the underlying sources and the deduction process may introduce additional errors. Henricksen, in [9], proposes a classification by dividing context information into four categories: data received from sensors, static data, profile data, and derived data. Context has a number of properties given the nature of context information that reflects its relative nature:

1. the context presents a series of temporal features, so depending on the evolution over time, the context may be static or dynamic. The static context consists of data that does not change over time, the dynamic context refers to information that changes permanently;

2. the context is imperfect: in many applications, the context is constantly changing, the data is getting too fast, and the decisions taken in the light of these data may be erroneous. In [10] four degrees of context imperfection are introduced: unknown data, ambiguous data, imprecise data, erroneous data;

3. context data interrelated with each other or can be inferred from other data;

4. the context has many alternative representations.

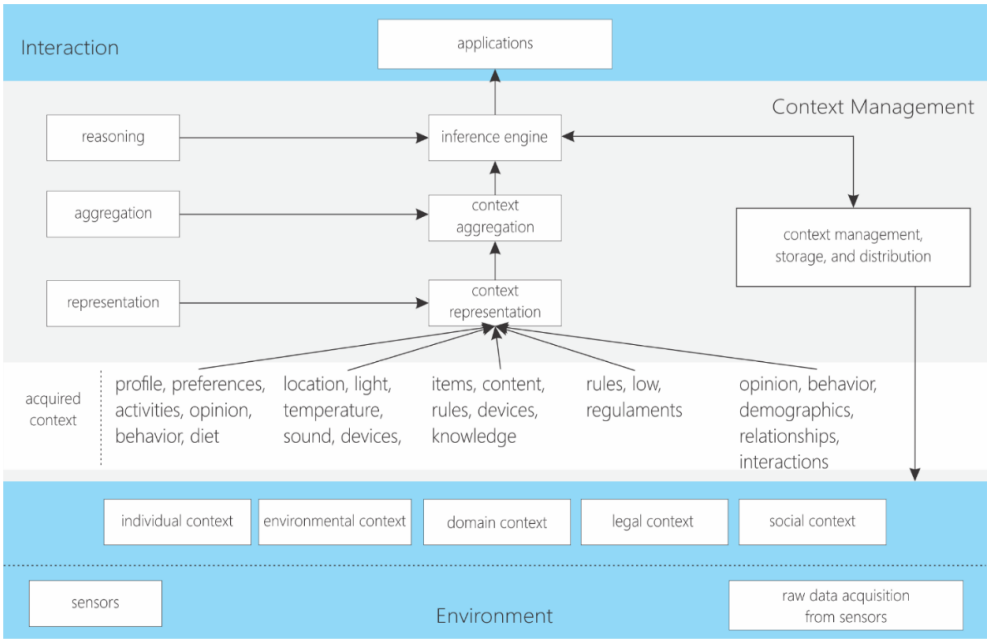

Figure 1. *The overall architecture of context-aware applications*

2.2. **Context-aware applications**

Context information is used in the interaction between the user and the application to enrich the app's data with information from the user environment or the analyzed entity. A context-aware application uses contextual information to reduce data explicitly entered by the user and to better respond to the user's requests. From [11], context-aware applications are: *"programs that adapt to where they are used, objects and people who are nearby, and their time changes"*. Dey, in [4], proposes a more generic definition, it says

that a system is aware of the context if "it uses the context to provide the user with relevant information and / or services, where relevance depends on the user's task". Dey's definition does not require applications to adapt to the context but to directly reflect context changes to the user, or to detect context changes – detection and interpretation can be performed by other entities. The overall architecture and components of context-aware applications in digital ecosystems can be observed in Figure 1.

In [12], three levels of context-awareness are distinguished:

1. personalization – this permits users to manually set their preferences and expectations, for example, the user sets the preferred temperature, and the system will keep that temperature in every room;

2. passive awareness – the system monitors and extracts data from the user and application environment, informs the user and offers the possibility to act, for example if the user enters a supermarket then the mobile phone or another device (such as a wearable) notifies the user of the products that have a discount available;

3. active awareness – the system automatically and continuously monitors the situation and acts autonomously, for example, if the system through the temperature and smoke sensors detects the presence of a fire in a smart house, the system can alert the owner and firefighters through various communication mechanisms.

3. *Context representation*

This research has explicitly focused on the importance of the context representation and manipulation, in the overall architecture of digital ecosystems. The context must be represented so that it can be understood and processed by the computer; the way the context is represented determines the mechanisms that can be applied to its processing. Context processing refers to the interpretation and aggregation of extracted context information, including a number of default properties that can be used to validate the context model. We present a series of context representation systems.

3.1. **Logical representation of context**

Under the logical representation of the context, this is an object that appears in a series of logical rules, in which a rule of $ist(c, p)$ expresses that the proposition p is **true** in context c, where c contains all the elements that are not explicit in p but which are necessary to correctly interpret the meaning of the sentence p. In this representation, given by John McCarthy [13], each context has associated a vocabulary in which the sentences are expressed. The sentence p can be expressed differently from one context to another and there are contexts in which this sentence cannot be expressed. In McCarthy's description (although we do not have a formal definition) we have the following properties of the context:

1. context cannot be described completely;

2. the context can be dimensionally infinitive;

3. any context is relative to another context;

4. if in a deduction we have more expressions in different contexts then there is a common context in which all expressions can be expressed in order to be interpreted in a unitary way.

A number of applications have been developed on the basis of this representation, which allows for the integration of several heterogeneous sources of information into a system and allows solving conflicts at a semantic level. For each data source, a context is automatically created, and then these contexts integrate into a new context where conflicts are resolved.

3.2. **Key-value representation of context**

Context modeling using key-value pairs introduced in [14] is used to construct a context-sensitive, mobile distributed application. The model captures three important aspects of the context by trying to answer three questions about users and resources: where you are, who you are and what resources are available nearby. The application uses a location browser to display relevant information based on the location of the users. This is a simple model commonly found in modeling distributed services, the model is not effective in modeling data with a more sophisticated structure. The Context-Toolkit [15] is an example of a framework that uses context representation as key-values that are encoded in XML for storage and transmission. The system needs a central server in which sensors, interpreting and aggregation modules are recorded to be discovered by applications. It is not enough to represent the context in digital ecosystems. The main disadvantages are: limited capacity to represent complex data types; is limited in the representation of relationships, dependencies, flows, and the quality of contextual information; does not allow verification of data consistency; does not provide support for deduction; does not solve cases of uncertainty; does not allow hierarchy, construction, and reuse of high-level context objects.

3.3. **Markup representation of context**

Context representation through markup languages organizes context information in a hierarchical way, uses various generic markup languages (SGML) variants such as XML and RDF/S by extending vocabulary defined in standards such as UAProf (User Agent Profiling Specification), which at its turn is based on CC/PP (Composite Capabilities/Preferences Profile – developed by W3C) [16]. The extension aims at using procedures that include the complex and dynamic nature of contextual information. Context elements are described by a set of profiles, each describing a different context. In [17] an evaluation of the CC/PP model is made and the Comprehensive Structured

Context Profiles (CSCP) is proposed as a flexible, decomposable, serializable and extensible model. CSCP does not define a fixed structure, the attribute names being interpreted according to their position in the profile structure (can be ambiguous at the profile level), CC/PP having a hierarchical structure on two levels with non-ambiguous attributes on the whole profile. CSCP is modular, extensible; it allows expression of user preferences and form expressions if-then-elsif-else.

3.4. **Graphical representation of context**

Graphical representations of the context are mostly used to structure the context (in describing the architecture of context-dependent systems [18] and less to manipulate its instances. Graphical representation languages use UML (Unified Modeling Language) or ORM (Object-Role Modeling) language extensions:

CML language: CML (Context Modeling Language), proposed by Henricksen and Indulska ([19], [20]), is an example of an ORM extension by adding new properties and possibilities of contextual classification of facts. Thus, the main element in this representation is fact (fact) and the modeling of the context rests upon identifying some types of facts and the ways in which various entities are involved in these facts. The model allows the classification of facts according to the source and their persistence, so we will have static facts (do not change and exist as long as there are entities involved) and dynamic facts (which in turn are divided according to source of origin in profiles, data obtained from sensors or derived information). CML introduces the possibility of describing a link between facts and dependence, through the *dependOn* relationship, through which we can specify how two facts are linked together, the link allows the change of one fact to induce changes in the other fact. The CML language allows us to include elements related to the history of a fact in context representation.

Context UML: Another way of graphical representation of the context proposed in [21], uses UML and a driven development approach to the development of context-aware web services (CAS). The authors present a series of meta-models for context, services, and mechanisms that associate services with context-inducing context-awareness. The methodology described in [21] used in modeling in the development of context-sensitive web services, allows the separation of modeling of context-shaping services. A number of mechanisms related to context perception have been abstracted and associated with relevant web services, thus inducing context sensitivity. The separate modeling of context and services facilitates the development, maintenance and expansion of CAS systems.

3.5. **Relational representation of context**

A relational contextual information representation model is introduced in [19] and [22] which allows the imposition of restrictions and is suitable for storing information in databases and can be easily interrogated by applica-

tions that need context data. Mapping context data into a relational system leads to a representation of abstract relationships in the form of relationships, and simple facts in the form of records stored in databases. Let R be a set of relations over the elements of a model C, if r $\in$ R we say that r [c_1, c_2, ..c_n] is true if the block <c_1, c_2, ..c_n> belongs to C. The *null* value can be assigned tuple elements, this value has been introduced to operate with unknown, uncertain and ambiguous situations. The value of the expression r [c_1, c_2, ..c_n] is unknown if there is ambiguity (several tuples are found) or if the tuple<c_1, c_2, c_n> is not present in the C but if we replace some components with the null value, we can find the tuple in the model. The situations are represented using predicate logic, in formula S (v_1, ...,v_n): φ, S represents the name of the situation, v_1, ..., v_n are variables, and φ is a logical expression in which the variables are exactly the ones in {v_1, ..., v_n}. Logical expressions can use the logical operators, OR, AND, NOT, equalities, inequalities and logical expressions of the form r [t_1, ...,t_n].

3.6. **Objectual representation of context**

Context modeling through object-oriented techniques brings the advantages of object-oriented modeling: legacy, encapsulation, and reuse. Existing implementations use various objects to represent different types of context data (such as location, temperature) and encapsulate the details of internal representation and context data processing. Access to context and some logic processing functions is enabled through a well-defined interface.

Java Context Awareness Framework: JCAF is a framework for developing context-aware applications in Java [23]. JCAF consists of a distributed set of context services, each service responds to a particular context, for example, each service can monitor one room in a smart home. The environment provides mechanisms for access to resources, journaling, and a transformation repository. Customers can access their entities and contexts in two ways, by query-response, by querying a particular entity and its data, or by subscribing/registering using the *EntityListener* class through which we can receive notifications for each change of context data from multiple entities. There is a subscription based on the type, so a client can subscribe to receive notifications from all entities of a certain type. The system can be adapted for use in digital ecosystems.

CORTEX: CORTEX [24] is an example of a middleware platform that enables the development of context-aware applications based on the Sentient Object Model – a mobile-based system that uses STEAM –, a special event-based location service specially designed for applications running on wireless networks. A "sentient" object encapsulates three main elements: Sensory Capture, Context hierarchy, and Inference engine. Communication with objects is done through a well-defined interface. They connect to the sensors and emit events to a series of actuators that alter the behavior of the program according to the received events. The "sentient" objects can connect in a tree array by constructing context hierarchies, so an object can play the role of sensor and

actuator for other objects. Inference engine is programmed under the CLIPS (C Language Integrated Production System). A graphical development tool is used to represent the context of an application that allows the representation of relevant sensors, the addition of actuators, the definition of the hierarchy of objects and the rules with which the inference engine will work.

Technology for Enabled Awareness: TEA system shown in [25] defines the context as the totality of information related to the state of the user and the application, including the location and the surrounding situation, it uses cues as an abstraction of the logical and physical sensors, each indication results from a single sensor, but by processing the data there may be more clues to a sensor. The use of the indices allows for a summary of the data extracted in a time interval and the application of some statistical functions (average, average square deviation, etc.) allows extraction of some data flow characteristics.

Hydrogen: The Hydrogen project shown in [26] proposes an application development framework that runs in various mobile devices with limited resurface. This framework has a three-state architecture: Layer Adapter – responsible for extracting sensor data, Management Layer – responsible for storing context data and sharing it with other devices nearby, and Application Layer – allows applications to access the lower layers. A distinction is made between the local and remote contexts. The main component of the management layer is the Context Server that stores context information and makes it available to all applications running on the device, communicating with other devices in a peer-to-peer manner. Obtaining context information from the context server can be done in two ways: by subscribing and by query. This model can be adapted and used in digital ecosystems.

Active Object Model: Active Object Model [27] is another approach in shaping the context that falls into this category. This model was the basis of the GUIDE project, a context guide for visitors to Lancaster, which aimed at integrating as wide a range of personal and environmental data as contextual information while maintaining system scalability. Context data and aggregation are performed through active, intelligent objects that adapt their behavior to visitors' preferences, but also to the current location. Objects expose a user interface and hide implementation details about data collection and initial processing. The user interacts with the application through a web browser that connects to a local web server, which in turn calls the GUIDE services through an object that extracts data from the sensors. As the visitor moves a resolver object, it generates instances of interest objectives for the visitor taking into account the location and its preferences extracted from an object representing the user's profile.

Ambient Components: As an extension of the object model we also find components of the context based on components. For example, in [28] the authors propose a component-based platform for extracting context data in which elements that abstracts both sensors and aggregation components are represented by a generic component called ambient component (similar to

Dey's widget context in [29]). The proposed architecture includes four layers and an application layer: the sensor layer, the fusion/aggregation layer, the assignment layer and the object repository (hive context). The model implies the existence of a repository as a high level context service that acts as an intermediary separating applications that use context information from context data capture mechanisms. This separation allows you to modify objects that capture the context and aggregate primary data without altering, recompiling, or even stopping applications that use the context. The model resembles Econtxt and it can be extended to model the context within digital ecosystems.

3.7. **Ontologies representation of context**

The context can be regarded as a special type of knowledge, which makes us analyze ways of representing knowledge through ontologies and methods of enriching context information through deduction. In [30], the definition of ontology is stated: "an ontology is an explicit, formal specification of a common conceptualization. Conceptualization refers to a model of a phenomenon, identifying the relevant concepts of that phenomenon. The specification is explicit because the concepts and constraints of their use are explicitly defined. The specification is formal because it can be read by the computer, which excludes the use of natural language. Conceptualization is shared (shared) because it captures a known knowledge, commonly accepted by a group, not an entity-specific information."

Context in the Gaia system: Gaia is a Meta operating system that builds as a distributed middleware that coordinates heterogeneous applications and devices in a physical space. Gaia is a system designed to support the development and running of portable and mobile applications that work in active spaces – programmable computing environments, where the user can interact with multiple devices and services simultaneously. Gaia's architecture includes three major components: kernel, application framework, and active space applications. The kernel contains core management component and a set of core interleaved services, services used by all Gaia applications: Event manager, Context service, Presence service, Space repository, Context File System. The Gaia system has been expanded into [31] by adding a Knowledge Base in which ontologies describing entities in the system and an ontology server that maintains a cumulative ontology for each space are loaded. The GAIA system is limited in security, no basic service within the system has security features.

CoBrA (Context Broker Architecture): CoBrA [32] is an agent-based system that uses ontologies to model the context in smart-space applications. The system uses a central agent (broker) to distribute context information to mobile agents. The context is a collection of ontologies describing the halls, events, people and their properties in a space used for meetings, presentations and conferences. The CoBrA system maintains a shared model of the entire community of agents, which can be applications installed in user-owned mobile devices, services running in conference room devices and web services

that maintain and share a situation with people, places, and objects in the real world. Basic ontology in COBRA-ONT is Place that abstracts a physical location, it contains a number of properties such as name, geographic position. Similarly, we have ontologies describing the agents' context, location and activity. The inference engine called F-OWL is a platform and a language-oriented object of knowledge representation.

Aspect-Scale-Context Information: The Aspect-Scale-Context Information (ASC) model introduced in [33] uses ontologies as a unit-specific method for specifying concepts, facts, and facilitating the sharing and reuse of context-related knowledge in sensor systems. Ontologies are described in CoOL (Context Ontology Language) introduced in [34] which is extended with elements used in the description of web services. The language allows to determine the interoperability and compatibility (and substitution possibilities) of services from a context perspective, it is used to implement context-awareness in various distributed environments and applications.

CONON: The CONON model introduced by Wang in [35] is similar to ASC, proposes a system of ontologies to represent the context in the same way, because of their ability to represent knowledge, facilitate the deduction, sharing and reuse of knowledge. The authors propose a high-level ontology called *ContextEntity* that retains the general aspects of context elements and a number of domain-specific ontologies (such as time, space, activities). The system writes and reads the ontologies described in the OWL-DL language (an extension that adds DL-descriptive logic-specific elements), language that allows consistency checking and deduction of knowledge using deduction engines created for logical description language systems.

SOCAM: The SOCAM (Service-Oriented Middleware-Oriented Services) introduced in [36] is an infrastructure to develop context-aware mobile services. The authors divide the application domain into different subdomains and define OWL ontology (according to the CONON model) for each subdomain to reduce the complexity of computers. The SOCAM architecture consists of the following components: Context Providers, Context Interpreters, Context Database, Service Location Service, and Context-aware Mobile Services. A context provider can deliver context (internally or externally) to the mobile service or other provider that can also provide high level context information to other services. The context is represented as an instance of a model described by an ontology. The most important component of the SOCAM system is its reasoning system, which consists of interpreters and the context database, it uses two types of ontologies: domain-specific and general ontologies. Each domain-specific ontology is connected to a general ontology and is reconnected with another general ontology whenever the context changes. The deduction system is modular; it allows coupling interpreters to perform different types of deductions. Centralized architecture makes it hard to use in digital ecosystems.

CoDaMoS: Within the CoDaMoS system (Context-Driven Adaptation of Mobile Services) [37] is proposed a very general model of the ontology-based

context used to model an intelligent environment (AmI). Multiple sets of ontologies are used to express context information about users, platform, services, and the environment. The user plays an important role in the smart environment, applications need to adapt to the user, profile, preferences, mood and current activity. The environment in which the user and application act is an important aspect of the context, the environment contains location elements, time and other environmental conditions such as temperature or lighting. The services provide the user with some functionality, the syntactic and semantic specification of the services makes it easy to discover and use them through well-defined interfaces. Each device contains its own context describing its own services and a series of pointers to the relevant information in other devices in the current environment. The system is flexible as SOCAM, there is no limit to the number of contexts that can be defined.

Context Managing Framework: The Context Managing Framework (CMF) introduced in [38] consists of four functional entities: context manager, resource servers, context recognition services, and applications. Context Manager is a central server that manages context information and delivers this information to client applications. Resource servers get context information from the various data sources they are connected to, then operate on these data (using fuzzy logic) and deliver data to the context manager. Context recognition services are used by the context manager to create high-level context objects based on atomic context data. The recognition system is modular and extensible, allowing for easy expansion by adding new recognition services. The model is based on distributed services, but uses a central server that manages and distributes context data, does not appeal to digital ecosystems.

3.8. **Hybrid context representation**

Hybrid models of the context use multiple representations from the ones presented, the Gaia model is considered by some authors to be a hybrid model that combines context-shaping elements with concepts from the operating systems domain. There are several hybrid model proposals that combine spatial context models with an ontology-based context representation. In [39], Bettini et al. present an analysis of context models, evaluate each model in part from the perspective of methods of expression (of formal expressiveness) and support for reasoning operations. It expresses the necessity of using hybrid models and proposes a multilayered hybrid context model that contains a sensor-based data extraction level, a level of representation of sensor data and aggregation, and a deduction based on ontologies. In [40], Henricksen et al. present a hybrid context model that combines the representation of CML with an ontology-based representation.

COMANTO: In [41] we have another hybrid model that combines existing patterns in a single model with ontology-based representation in an attempt to gain the benefits of existing models, but also the expressiveness and support for deduction offered by OWL and ontologies. Other context models (considered traditional) process context information in a similar way to databases.

In order to achieve the advantages of both approaches, a hybrid, scalable context model with efficient deduction and interpretation mechanisms is proposed in [41], which can be used in a large-scale, context-aware distributed system. The context model known as COMANTO (COntext MAnagement oNTOlogy) uses a context-based (location-based) spatial model and describes generic context types that are not dependent on a specific domain or a particular situation. The spatial space model addresses data management issues in context-aware distributed systems; it is integrated with the knowledge base of the COMANTO system resulting in a model that can be applied to many applications.

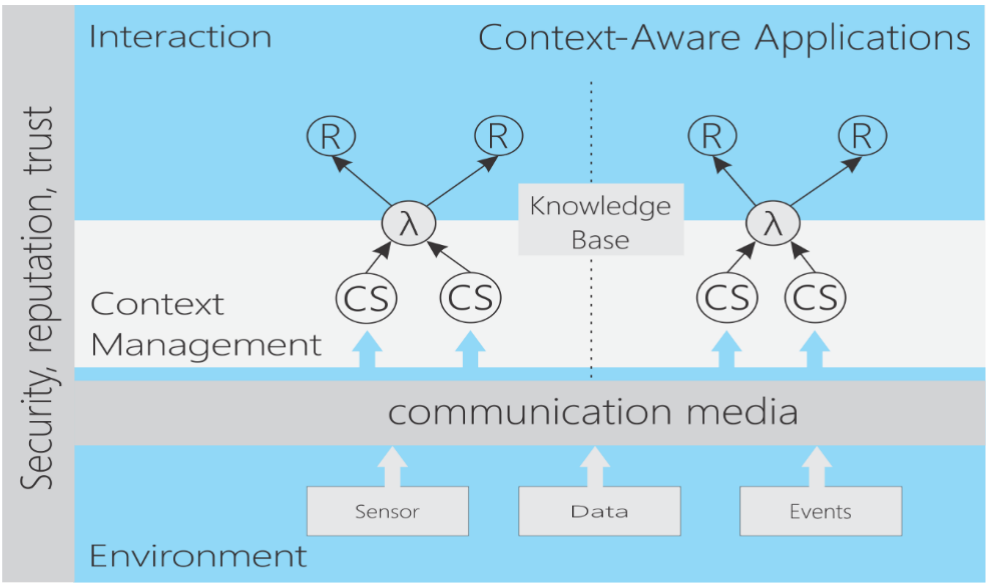

Figure 2. *Econtxt architecture*

3.9 **Stream representation of context**

We introduce the Econtxt architecture in a nutshell – the programming model of context for applications running in a digital ecosystem proposed in [42]. The context assumes the existence of a medium (of communication) where different entities provide context data of different types. Data is extracted through *consource* objects (CS) which listen for a certain type of context data. Any entity issuing that type of data is automatically detected, context appears as a stream of data, where every piece of information contains the identity of the issuer. Data streams can be processed in a variety of ways and can be combined to get high-level context data. In addition, context data is automatically implicated, in each update, in recalculating values of "reactive" variables (denoted by R) that are used in the application. The application uses the reactive variables that have an updated value at any time. The overall architecture of Econtxt can be observed in Figure 2.

4. *Discussion*

All systems considered follow an architecture that abstracts how to obtain the context, thus separating the capture of context data from the other areas of the application. The technology by which sensor data is extracted is different in each frame, but it is encapsulated in different components and separated from the rest of the application.

There is no standard for data mining, so we have to deal with many proprietary implementations. SOCAM uses the most sophisticated approach to extracting context data, environmental data streams that shape the context in the Econtxt model are more suited to digital ecosystems.

The context model and deduction systems are another aspect taken into account in the construction of context-aware applications. Ontologies are a tool for specifying context information, starting from ontologies, a deduction system can retrieve new information and adapt the behavior of the application accordingly. Context Toolkit, for example, is easy to understand but has a simple context model based on unmatched key-value pairs, and deduction operations cannot be applied. SOCAM uses a context model based on ontologies; it defines a general ontology and allows the definition of specific ontology.

The mechanisms of discovery of sources are less exploited in the analyzed frameworks. Such mechanisms are especially necessary in pervasive systems where context data sources, sensors, often change. In Econtxt, data sources of the same type are automatically detected.

The ability to retain context history data is another important aspect that is worth considering. History allows the use of learning algorithms that can be used to construct intelligent, context-sensitive services. In addition, based on history, context information can be approximated in advance, and applications can proactively provide the user with a number of useful services. Some systems store historical data but do not use learning algorithms. In Econtxt context data comes in the form of a stream, various data aggregation operations can be applied to the stream.

Another important aspect is data security. Context-aware systems often manipulate data related to user profile and preferences. This requires policies and mechanisms to ensure the protection of sensitive data. The CoBra system uses a language that describes context access policies. Gaia uses several mechanisms to introduce access restrictions and secure communications on user locations. The Context Toolkit defines a context ownership that determines who has access to context data.

Explaining the context in the form of key-value pairs is the easiest way to represent context data, easy to use for small sets of data. The model is not scalable and does not allow the representation of complex or hierarchical relationships. It is used in simple applications for managing configuration data or user preferences.

Representing context data using markup languages allows for efficient data mining, data can be validated through definitions schemas. Instruments for the processing and validation of XML data are available. It does not provide the necessary advanced expressivity in deduction operations. In the

absence of standards (or universally accepted medals), context modeling, data mining, reuse and interoperability are difficult. It is used in many cases to represent the user profile.

Model	Architecture	Sensors	Context model	Deduction engine	Resource discovery	History	Security
CoBra	Agent based	Context acquisition module	Ontologies (OWL)	Inference engine and knowledge base	-	Available	policy language
Context Management Framework	Blackboard based	Resource servers	Ontologies (RDF)	Context recognition service	Resource servers and subscription mechanism	-	-
Context Toolkit	Widget based	Context widgets	Attribute-value tuples	Context interpretation and aggregation	Discoverer component	Available	Context ownership
CORTEX	Sentient object model	Context component framework	Relational data model	Service discovery framework	Resource management component framework	Available	-
TEA	Local cues	Cues	Object oriented	Cue base framework	-	-	Custom
Gaia	MVC (extended)	Context providers	4-ary predicates (DAML + OIL)	Context-service module (first-order logic)	Discovery service	Available	Supported
Hydrogen	Three layered architecture	Adapters for various context types	Object-oriented	Interpretation and aggregation of raw data only	-	-	-
SOCAM	Distributed with centralized server	Context providers	Ontologies (OWL)	Context reasoning engine	Service locating service	Available	-
JCAF	Distributed and Cooperating Services, P2P	Context class	Object oriented	Transformers	Subscription to context container	logging facilities	Access Control
Ambient Components	Ambient Component based platform	Abstract sensors	Ambient Component	Aggregation taxonomy	Object repository	Context hive	-
Aspect Scale context information	Distributed platform	Sensors system	Context Ontology Language	Ontology based deduction	-	-	-
CONON	ContextEntity	Abstract sensors	OWL-DL described ontologies	Ontology based deduction	-	-	-
CoDaMoS	Intelligent ambient	Context Providers	Ontologies (OWL)	Ontology based deduction	-	-	-
COMANTO	Hybrid, distributed	Semantic Context Entity)	OWL ontologies	Ontology based deduction	-	-	-
ECONTXT	Consource	Abstract sensors	Stream oriented	Aggregation of raw data, update of reactive data	Automatically type based	Stream history	Custom

Figure 3. *Evaluating context models*

The graphical representation of context data through UML or ORM provides a better expressivity than the key-value representation; these models allow relationships to be represented in the context. The storage of context data can be done using SQL databases, noSQL, XML databases, and more. Databases are easy to use, can store large amounts of data, SQL language allows for complex queries.

Objectual representation of context data uses class hierarchies and their relationships, objects promote encapsulation and reuse. The context can easily

Representation	Advantages	Disadvantages	Applications
Key-Value	Simple. Flexible. Easily managed.	Not scalable, dependent on applications, doesn't have a solid structure. On larger sets of data, extraction and deduction become difficult operations. Doesn't offer support for the representations of complex relationships. There is no validation support and no standardized tools.	Can be used on limited, independent data, user preference modeling, and application configurations.Used to store and transfer data on less complex cases.
Markup languages, XML	Flexible, more structured.Validation is possible via schemas.There are many processing tools.	The extraction of information has a moderate complexity. The system can become complex if information is on more levels. Dependent on applications, no standard for data sorting.	Allows the components of a system and data structures to be decoupled. It is used for network data exchange.
Graphic representation, databases	Validation is possible through constraints. It allows the modeling of complex relationships. Data extraction is relatively easy. There are different standards and implementations.	May require special configurations.Querying can become complex.Low interoperability between different implementations. It is guided by design principles, it has no standards.	Suitable for handling large quantities of data and can also be used for long-term storage.
Object representation	It allows the modeling of complex relationships. There are processing tools, easy to integrate through programming languages.	Missing validation methods.It is guided by design principles, it has no standards.Extraction of information is difficult.	Allows context data to be represented at the program code level. Provides runtime manipulation to context data typically stored in memory.
Based on logic	Simple, easy-to-use model. Allows generating new high level context data from primary context elements. Allows deduction rules to apply. There are a number of processing tools.	Lack of standards. Missing validation methods. Has a strong engagement with applications.	It can generate new information based on primary context data. Defines restrictions and constraints, shapes events and actions (executed at event detection).
Based on ontology	It is expressive. Allows semantic deductions. Admits validation mechanisms. It allows data sharing, it is independent of applications. It is standardized, there are processing tools.	Representation can become complex.Data extraction and deduction becomes a complex and resource-computing process.	Representation can become complex. Data extraction and deduction becomes a complex and resource-computing process.
Representation throw data flows	Permits composite distributed sources, data aggregation, historical. Allows you to add new independent application sources.	It has no inference engines, they can be added however. Participants are considered trustworthy. Security is provided at the network level.	Context data appears as logs that can be stored in databases, logical deductions, data mining algorithms, etc. can be applied.

Figure 4. *Comparison of context modeling systems from the perspective of data representation*

integrate into written applications using object oriented languages that can model internal mechanisms for extracting, manipulating, and storing context data.

Logical representation of context data uses facts, expressions and rules, rules are used to express constraints, preferences, or policies. Provides increased expressivity and facilitates deduction operations; lack of standards makes application harder and more reusable. Ontologies aim to reuse common knowledge, allow the integration of knowledge from different domains into applications. They are generic, provide high expressivity, versatility, validation and extensibility. These allow work with incomplete information and uncertainty.

In Figure 4, we present a summary of the advantages and disadvantages of different context modeling systems and the applications in which they can be used, depending on how the internal context information is represented.

A comparison of the context models from the perspective of deduction techniques used follows, after an analysis from [12].

	Advantages	Disadvantages	Applications
Deduction based on rules	Easily defined system, easily expandable, doesn't consume many memory and calculating resources.	A manual definition is used that can lead to errors. We do not have validation or verification systems.	It can be used in cases where primary data is aggregated in high-level context data. Used to define events and actions.
Ontologies	It provides a complex representation and allows the use of sophisticated deduction algorithms that produce rich signage results. I can count both numeric data and text. It is possible to validate and verify the correctness.	Data must be in OWL or RDF format. It has a limited numerical porting. Low performance.	It is used in applications where the structure and meaning of the data is critical.
Probabilistic logic	They can operate with new, unpredictable situations. Various models are available: Byes, Markov. They can manage uncertainty, produce results with moderate semantics.	Accepts only numeric data. Requires prior knowledge of probabilities.	Applies in cases where context data come from various sources with a known probability.
Unsupervised learning, clustering	It does not require data for learning, it is not necessary to know the output in advance.	Models are complex, difficult to validate, results are poor, unpredictable. There are computational resource-consuming systems.	It is used in cases where the results are not known in advance.
Logic (fuzzy)	Easily definable, allows a natural representation, allows expansion, allows for uncertainty management. It does not require large computing resources	Easy to define, allows for natural representation, allows expansion, allows for uncertainty management. It does not require large computing resources	It can be used in cases where primary data is aggregated into more "natural".
Supervised learning, decision trees, neural networks, etc.	There is a theoretical basis. There are several models available. Good accuracy.	Requires large amounts of numeric data. It requires training data. Selecting a representative set of data could pose a difficult task. The deduction process is a consumer of computing and memory resources. Models can be complex, do not easily mold the existing knowledge.	Applies in cases where data can be easily identified, the possible output is known. Data is numeric and large.

Figure 5. *Comparison of context modeling systems from the perspective of deduction techniques*

5. *Conclusions*

In this article we have presented a series of approaches to modeling, extracting and manipulating context information in various types of context – aware applications. Some of them are also used for applications within digital ecosystems. The architecture of such an application depends largely on how to acquire context data. All systems considered follow an architecture that abstracts how to obtain the context, thus separating the capture of context data from the other areas of the application. There are several prospective directions for future work that should be explored in modeling, extracting and distribution of context informations in digital ecosystems. Context representation and reasoning methods must be studied in relation with context acquisition and situation recognition tasks, which may determine the correct representation mode and the aim of the reasoning processes. This work follows the study conducted in [43] and continues the research that was presented in the respective paper.

Acknowledgements. *The work has been funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/132395.*

References

1. M. Weiser, *The Computer for the 21st Century,* "ACM SIGMOBILE Mob. Comput. Commun. Rev.", vol. 3, no. 3, pp. 3–11, 1999.
2. R. W. Schilit, B., Norman Adams, *Context-Aware Computing Applications,* 1994.
3. H. Koç, E. Hennig, S. Jastram, and C. Starke, *State of the Art in Context Modelling – A Systematic Literature Review,* in "Advanced Information Systems Engineering Workshops SE - 5", vol. 178, L. Iliadis, M. Papazoglou, and K. Pohl, Eds. Springer International Publishing, 2014, pp. 53–64.
4. A. K. Dey, *Understanding and Using Context,* "Pers. Ubiquitous Comput. J.", vol. 1, no. 5, pp. 4–7, 2001.
5. G. Chen and D. Kotz, *A Survey of Context-Aware Mobile Computing Research,* "Dartmouth Comput. Sci. Tech. Rep.", vol. 3755, pp. 1–16, 2000.
6. M. Bazire, M. Bazire, P. Brézillon, and P. Brézillon, *Understanding Context Before Using It,* Model. Using Context 5thInternational Interdiscip. Conf. Context 2005, Paris, Fr. July 5-8, 2005, vol. 3554, pp. 29–40, 2005.
7. P. Brézillon, *Context in Problem Solving: A Survey,* "Knowl. Eng. Rev.", vol. 14, no. 1, pp. 47–80, 1999.
8. L. D. Serbanati, A. Vasilateanu, and B. Nita, *Strengthening Context-Awareness of Virtual Species in Digital Ecosystems,* in Control Systems

and Computer Science (CSCS), 2013 19th International Conference on, 2013, pp. 503–510.
9. K. Henricksen, (Thesis) *A Framework for Context-Aware Pervasive Computing Applications*, Queensland: University of Queensland, 2003.
10. K. Henricksen and J. Indulska, *Modelling and Using Imperfect Context Information,* Second IEEE Int. Conf. Pervasive Comput. Commun. Work. Context Model. Reason. (CoMoRea', vol. 4, pp. 33-37.
11. B. N. Schilit and M. M. Theimer, *Disseminating Active Map Information to Mobile Hosts,* "Network", IEEE, vol. 8, no. 5, pp. 22–32, 1994.
12. C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, *Context Aware Computing for the Internet of Things: A Survey,* "IEEE Commun. Surv. Tutorials", vol. 16, no. 1, pp. 414–454, 2014.
13. J. McCarthy, *Notes on Formalizing Context,* pp. 555–560, 1993.
14. B. Schilit, N. Adams, and R. Want, *Context-aware computing applications,* in Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on, 1994, pp. 85–90.
15. D. Salber, A. K. Dey, and G. D. Abowd, *The Context Toolkit?: Aiding the Development of Context-Enabled,* CHI '99 Proc. SIGCHI Conf. Hum. Factors Comput. Syst., pp. 434–441, 1999.
16. W3C Working-Group, *CC/PP ver.2.0,* 2010. [Online]. Available: http://www.w3.org/TR/CCPP-struct-vocab2.
17. J. Indulska, R. Robinson, A. Rakotonirainy, and K. Henricksen, *Experiences in Using CC/PP in Context-Aware Systems,* in "Mobile Data Management", 2003, pp. 247–261.
18. K. Henricksen and J. Indulska, *A Software Engineering Framework for Context-Aware Pervasive Computing,* Proc. PerCom'04, 2004.
19. K. Henricksen and J. Indulska, *Developing Context-Aware Pervasive Computing Applications: Models and Approach,* "Pervasive Mob. Comput.", vol. 2, no. 1, pp. 37–64, 2006.
20. Q. Z. Sheng and B. Benatallah, *ContextUML?: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services,* 2005.
21. K. Henricksen, J. Indulska, and A. Rakotonirainy, *Generating Context Management Infrastructure from High-Level Context Models,* In "4th International Conference on Mobile Data Management (MDM)-Industrial Track", 2003.
22. J. E. Bardram, *The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications,* "Pervasive Comput.", no. April, pp. 98–115, 2005.
23. G. Biegel and V. Cahill, *A Framework for Developing Mobile, Context Aware Applications,* 2nd IEEE Conf. Pervasive Comput. Commun. PerCom 2004, vol. 26031, 2004.
24. A. Schmidt and K. Van Laerhoven, *How to Build Smart Appliances?,* "IEEE Pers. Commun.", vol. 8, no. 4, pp. 66–71, 2001.
25. T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, and W. Retschitzegger, *Context-Awareness on Mobile Devices – the Hydrogen Approach,* vol. 43, no. 7236, 2002.

26. K. Cheverst, N. Davies, K. Mitchell, and A. Friday, *Design of an Object Model for a Context-Sensitive Tourist Guide,* pp. 1–4, 1999.
27. C. Jacquet, Y. Bourda, and Y. Bellik, *A Component-Based Platform for Accessing Context in Ubiquitous Computing Applications,* "J. Ubiquitous Comput. Intell.", vol. 1, no. 2, pp. 163–173, 2007.
28. A. K. Dey, *Providing Architectural Support for Building Context-Aware Applications,* no. November, 2000.
29. S. Chen and M.-A. Williams, *Learning Personalized Ontologies from Text: A Review on an Inherently Transdisciplinary Area,* in "Personalized Information Retrieval and Access: Concepts, Methods and Practices", IGI Global, 2008, pp. 1–29.
30. A. Ranganathan, R. E. McGRATH, R. H. Campbell, and M. D. Mickunas, *Use of Ontologies in a Pervasive Computing Environment,* "Knowl. Eng. Rev.", vol. 18, no. 3, pp. 209–220, 2003.
31. H. Chen, T. Finin, and A. Joshi, *An Ontology for Context-Aware Pervasive Computing Environments,* "Knowl. Eng. Rev.", vol. 18, no. 3, pp. 197–207, 2003.
32. T. Strang, *Service Interoperability in Ubiquitous Computing Environments,* PhD thesis, Ludwig-Maximilians-University Munich, Oct. 2003.
33. T. Strang, C. Linnhoff-Popien, and K. Frank, *CoOL: A Context Ontology Language to Enable Contextual Interoperability,* "Ifip Int. Fed. Inf. Process.", vol. 2893, pp. 236–247, 2003.
34. X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung, *Ontology Based Context Modeling and Reasoning Using OWL 3. CONON?: The Context Ontology,* Proc. Second IEEE Annu. Conf. Pervasive Comput. Commun. Work., 2004.
35. T. Gu, H. K. Pung, and D. Q. Zhang, *A Service-Oriented Middleware for Building Context-Aware Services,* "J. Netw. Comput. Appl.", vol. 28, no. 1, pp. 1–18, Jan. 2005.
36. D. Preuveneers et al., *Towards an Extensible Context Ontology for Ambient Intelligence,* "Ambient Intell.", vol. 3295, pp. 148–159, 2004.
37. P. Korpipaa, J. Mantyjarvi, J. Kela, H. Keranen, and E. J. Malm, *Managing Context Information in Mobile Devices,* "IEEE Pervasive Comput.", vol. 2, no. 3, 2003.
38. C. Bettini et al., *A Survey of Context Modelling and Reasoning Techniques,* "Pervasive Mob. Comput.", vol. 6, no. 2, pp. 161–180, Apr. 2010.
39. K. Henricksen, S. Livingstone, and J. Indulska, *Towards a Hybrid Approach to Context Modelling, Reasoning and Interoperation,* "Adv. Context Model. Reason. Manag.", pp. 54–61, 2004.
40. I. Roussaki, M. Strimpakou, C. Pils, N. Kalatzis, and M. Anagnostou, *Hybrid Context Modelling: A Location-Based Scheme Using Ontologies,* Proc. Fourth Annu. IEEE Int. Conf. Pervasive Comput. Commun. Work., 2006.
41. J. Barwise and J. Perry, *Situations and Attitudes,* "J. Philos.", vol. 78, no. 11, pp. 668–691, 1981.

42. A. Averian, *A Programming Model of Context-Aware Applications in Digital Ecosystems,* 17 Int. Multidiscip. Sci. GeoConference 2017, vol. 17, pp. 37–44, 2017.
43. A. Averian, *Towards More Context-Awareness in Reactive Digital Ecosystems,* in Creativity in Intelligent Technologies and Data Science: Second Conference, CIT&DS 2017, Volgograd, Russia, September 12-14, 2017, Proceedings, A. Kravets, M. Shcherbakov, M. Kultsova, and P. Groumpos, Eds. Cham: Springer International Publishing, 2017, pp. 640–654.