Taylor & Francis
Taylor & Francis Group

Applied Artificial Intelligence
An International Journal

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uaai20

Contexts and Context-awareness Revisited from an Intelligent Environments Perspective

Juan Carlos Augusto

To cite this article: Juan Carlos Augusto (2022) Contexts and Context-awareness Revisited from an Intelligent Environments Perspective, Applied Artificial Intelligence, 36:1, 2008644, DOI: 10.1080/08839514.2021.2008644

To link to this article: https://doi.org/10.1080/08839514.2021.2008644

Published online: 02 Mar 2022.

APPLIED ARTIFICIAL INTELLIGENCE
2022, VOL. 36, NO. 1, e2008644 (725 pages)
https://doi.org/10.1080/08839514.2021.2008644

Taylor & Francis
Taylor & Francis Group

Contexts and Context-awareness Revisited from an Intelligent Environments Perspective

Juan Carlos Augusto

Research Group on Development of Intelligent Environments, Department of Computer Science, Middlesex University London, London, UK

ABSTRACT

Context is a useful concept somehow unconsciously used by humans in daily life problem-solving. Recently, several subareas of computer science have been increasingly trying to rely on this concept to design systems with practical use in certain predefined circumstances. The perception is that imbuing in the system certain context-awareness qualities can support intelligent decision-making in specific practical situations. Despite a significant number of implemented systems that aim at providing context-awareness, there is a lack of commonly accepted and used methodologies and tools. At the root of this, is the lack of agreement on a set of good principles or standards, which can act as a guide to the scientific community and the developers interested in this class of systems. There have been some extensive surveys on the use of context, still there is no theoretical corpus emerging that we can use to discuss the essential concepts making up the fabric of contexts and its use by system developers. Here we attempted such enterprise at a level, which is more formal than popular surveys, in a way that is not implementation dependent and in a way that highlights key concepts of relevance to developers. We reassessed first the basic concepts identifying the need to more prominently consider system beneficiaries' satisfaction. We then transfer explicitly these values to a more formal outline of the basic componentgs and the operations which emerge as relevant. We identify and highlight the tasks of context activation, comparison, influence, construction, and interaction. We hint at how these may work in practice and explained these through examples. We show how the theory is flexible enough by generalizing it to multiusers so that optimization of global preferences and expectations is used to drive system development and system behavior.

ARTICLE HISTORY
Received 24 May 2021
Revised 13 November 2021
Accepted 15 November 2021

Introduction

Computer Science has been continuously evolving since its inception less than a century ago. The last decades stimulated a more intimate connection with society at large and a spreading of computing resources everywhere in a way that increasingly affects our everyday life. This surge in the availability of

CONTACT Juan Carlos Augusto J.Augusto@mdx.ac.Uk Research Group on Development of Intelligent Environments, Department of Computer Science, Middlesex University London, London, UK

technology has led to a number of areas: first Ubiquitous Computing, "Ubicomp" (Weiser 1991), also termed Pervasive Computing and Communications or "Percomm," then Internet of Things, "IoT" (Atzoria, Iera, and Morabito 2010; Perera et al. 2014; Pradeep et al. 2021), Ambient Intelligence, "AmI" (Aarts and Roovers 2003) and Intelligent Environments, "IE" (Callaghan et al. 2000), (Hagras et al. 2004) and (Augusto et al. 2013). There are different nuances in each area and still, they share far more than what differentiates them. They are all intimately related by a core theme: context-awareness. The success of systems in these areas heavily depends on how well they are perceived by users to deliver their services when and how it matters. See for example (Stavropoulos, Vrakas, and Vlahavas 2013) and (Forbes, Massie, and Craw 2020). They tend to be highly specialized and can be seen as applied Artificial Intelligence (AI) systems, as all they nourish from simplified variations of AI algorithms and techniques for automated reasoning and learning (Augusto et al. 2017). The availability of new technology opening up opportunities for societal innovation has created many prototypes and commercial products. However, this throughput quantitative success has not led to emerging good practices, standards, or methodologies, which can be replicated in the traditional and successful manner Computer Science has built on in the past. It is as if fascination with the new possibilities occluded community reflection and discussion, which is holding back maturity in this sector. Theory, methodologies and tools that were good to build previous generations of systems are still somehow relevant, however, not necessarily optimal for systems which now have different components, emphasis and priorities. One such concept which needs a closer examination is context and its by-product of context-awareness. Initial considerations about context has been made in AI, as part of the Knowledge Representation and Reasoning (KRR) area (see Section 2), and later on borrowed and rewritten at the turn of the century in a less ambitious and more utilitarian way within emerging areas such as Ubiquitous Computing. Most influential was Dey's work (Dey 2001; Dey and Abowd 1999) which were instrumental on bringing this concept to the attention of the emerging areas and since then profusely cited. Dertouzos' reminder (Dertouzos 2002) of the need to keep humans at the center of our technological explorations has been influential on revisiting the more data and system centric predominant view brought by Dey's earlier definitions of Context and its derivation on Context-awareness. Here we follow (Dertouzos 2002) and (Augusto et al. 2013) on the line of thought that humans have to build systems for humans' sake not for technology sake, and that this is particularly meaningful to systems under the umbrella of Intelligent Environments. As a result of the generality of the prevailing context definition, the word context has been largely used by colleagues and developers in a way which is not fundamentally different from previously useful concepts such as "data" or "information." Complementary to the data-system conception of

contexts there has been very little community reflection and methodology creation to support developers. This article aims at addressing these two long standing issues. First we enrich the view of context and context-awareness with an explicit acknowledgment of humans. Then we use this updated understanding of those fundamental concepts to give some first steps in the direction of a more methodological understanding of this field and above all to start creating conceptual tools for developers. We do so with a mixed background of Artificial Intelligence and Software Engineering, challenged through more than a decade of developing practical systems, which are capable to work in the real world. This article contributes to the building up of bridges between AI and IE, a need identified in (Augusto et al. 2017). In doing so we mostly remain at a conceptual and theoretical level of analysis laying in between the more specific theories of contextual analysis as explored in AI and the most practical and implementation oriented IE approaches. We start by looking at relevant explorations of the notions of context and context-awareness (Section 2), then we reformulate these concepts rebalancing priorities by including the human in the theory (Section 3). This setting allows us to explore some alternatives for context-related operations (Section 4) from a new perspective. It also provide the basis to redefine Intelligent Environments in a way which links beneficiaries and context-awareness at application level with the overall system behavior (Section 5).

Through that content we revisit and explore new concepts in the direction of a body of knowledge which can better support developers to think about the systems they build, as well as a more systematic organized discussion within relevant academic communities.

State of the Art

There has been a long-standing interest, although rather marginal compared to other topics, in the understanding of the concept of "context" within AI (Guha 1991; McCarthy 1993; Giunchiglia 1993). More intensive use and with a more practical purpose of context was then pursued within the then emerging Ubiquitous Computing (Weiser 1991) and its latest ramifications. An analysis of these two areas Ying-Yang style interaction with the concept of context is offered in (Augusto et al. 2017). We are not going to replicate that comprehensive survey here, instead a few representative publications will be considered to illustrate what is the current need. A significant exploration of the concept from the KRR-AI perspective came from Trento as a continuation from Giunchiglia's work through a number of interconnected theoretical analysis. For example, in (Benerecetti, Bouquet, and Ghidini 2000) the metaphor of a "magic box" is introduced to argue that the mechanisms of contextual reasoning proposed in the literature can be classified into three general forms: localized reasoning, push and pop, and shifting. They associated these

three to notions to what they believe are three fundamental ways for context-dependent representations: "partiality" (the portion of the world considered), "approximation" (the level of detail at which the portion of the world is considered) and "perspective" (the point of view from which the world is observed). The authors distil two general principles of a logic of contextual reasoning which regulate the relation between models and contexts in the theory. Then (Benerecetti, Bouquet, and Bonifacio 2001) highlighted that the traditional view of centralized context was not sufficient to capture more complex systems where contexts are distributed and they may have its own internal notion of what context is as well as some need to interact with other contexts with different notion of contexts. Bouquet et al. (2003a) classified the main approaches to context within the Knowledge Representation and Reasoning area as grouped into two main strategies: divide-and-conquer which "sees context as a way of partitioning a global model of the world into smaller and simpler pieces," and compose-and-conquer which "sees context as a local theory of the world in a network of relations with other local theories." They prove although both were largely equivalent there were slight practical advantages for the second option.

The Magic Box metaphor can be useful to some extent to think about some context-related aspects of sound meaning transferring across contexts; however, practical context are much richer than that. The authors mainly deal with one way of obtaining knowledge; however, IEs do not necessarily stick to one way of making decisions in a system, usually they have to combine different contributions: data coming from sensors (most likely preprocessed through middleware), knowledge which has been acquired through learning algorithms, and continuous real-time decisions based on a combination of the former ones. Rather than the system understanding whether on the 1st of the month rained and today is the 2nd of the month then "yesterday rained" is appropriate from a linguistic perspective, contexts are preset (by design or by learning) and the system is continuously checking context detection, for detected contexts it will trigger pre-set actions (for example, if the room is dark and someone walks in then turn the light on). These practical contexts are difficult to put in a box, some contexts may be neatly separated, some share features, some are combined to make higher level ones, some information of the context may be semantically "blurred" by deteriorated sensing data, new contexts may be learnt (perhaps imperfectly first and refined continuously as a system revise its own knowledge).

There was a large impasse after the flurry of publications previous and around the turn of the century. Later worked is much more scattered and disconnected. There were attempts to connect the diversity of areas which use the concept of context (Bazire and Brézillon 2005) and also state of the art analysis to connect emerging ideas (Bettini et al. 2010). However, the diversity of general areas and ideas arising from very specific implementations

conspired for the community as a whole to agree on commonly accepted approaches considered to be good practice, and the golden standards of the area.

Siewe, Zedan, and Cau (2011) algebraic approach extended a preexisting the Calculus of Mobile Ambients (MA) by (Cardelli and Gordon 1998) into the Calculus of Context-aware Ambients (CCA): "In CCA the notion of ambient, inherited from MA, is the basic structure used to model entities of a context-aware system such as: a user, a location, a computing device, a software agent or a sensor." CCA follows Dey's definition of context and in view of the authors "Context-awareness requires applications to be able to sense aspects of the environment and use this information to adapt their behaviour in response to changing situations. These aspects of the environment that can influence the behaviour of an application constitute the context of that application." This again emphasized a system centered view of context and context-awareness. A diversity of context relevant entities are accommodated into the specific linguistic options offered by the process algebra (mostly inherited from MA), which is good for the theory of the algebra however could be a restricting factor for real applications in not representing different properties which differentiate entities. For example: representing a beneficiary, a phone, a room, with the same CCA system elements does not mean in real life are the same level and our industry needs to reflect the richness of those entities in real applications to properly capture the subtleties that makes them perceived as useful by reacting appropriately in appropriate conditions. There are also other issues with the absence of explicit time handling and of sensor data instability which are so fundamental in practical applications.

The multicontext systems (mMCS) by (Brewka et al. 2011) is a representative of what (Bouquet et al. 2003a) called "compose-and-conquer" approach to contexts and provides a KRR style reactive formalism suitable for continuous reasoning in dynamic environments, including abstract sensors, runs, and online reasoning can be addressed. Contexts where assumed to have a single logic rather than a logic suite and that management functions are deterministic. The theory considers contexts, their inner functioning and also operators at the higher "context management" level for example, so called "bridge rules." This represented a good evolution of the KRR two decades long discussion on how to organize the handling of contexts internally within the AI system. And it currently has evolved through (Ellmauthaler and Pührer 2015; Brewka et al. 2018; Dao-Tran and Eiter 2017; Beck, Dao-Tran, and Eiter 2018; Cabalar et al. 2019) into what can be considered the current state of the art theory within KRR.

Before transitioning to an updated understanding of contexts, we highlight the two main issues of concern which we explore in this article and which were not well represented in the work described above. One issue is the adequacy of the theories to the human-centric practical applications the area of Intelligent

Environments demands today, where intelligence of the system is measured by the practical results, the services expected by the main user. The theories above have a consistent focus on the artificial side of the system without acknowledgment of humans as an important part of the system (which should be reflected in the theory). We are not advocating for a full representation of humans with their knowledge, beliefs, emotions and so on, but only a minimal selection of human characteristics (e.g., preferences) which may be relevant and noticeable to the context-aware system and the relationship between the artificial system and the human entities interacting with it. The other big issue is the lack of a theory that goes beyond a generic definition and explores concepts really useful to developers creating real systems to foster debate within the academic community on how to revise and update methodologies in a way they are more relevant to the latest generation of systems. Previous formal work focused on what can be inferred within a context (where certain set of premises *P* are valid) and another (where certain set of premises *P*' are valid) and what can or cannot be shared amongst them (see Figure 1). Of course understanding that is useful, however, to create real systems in the real messy world with the technological availability of today, those theoretical explorations leave too many questions unanswered for developers who have to engineering modern real systems. Whilst those were interesting genuine philosophical endeavors beneficial to the theory of AI, IE developers are more preoccupied with questions such as: Which contexts are needed to achieve the expected system services? How these contexts relate to each other? When a context is activated? Which actions should be triggered, to whom, when, in which way? How can contexts be engineered with more predictable outcomes?

So this article is not about a theory of how a human reason about contexts in daily life neither is about how an artificial system can mimic that. Rather it is about how contexts can be developed in such a way that a context-aware

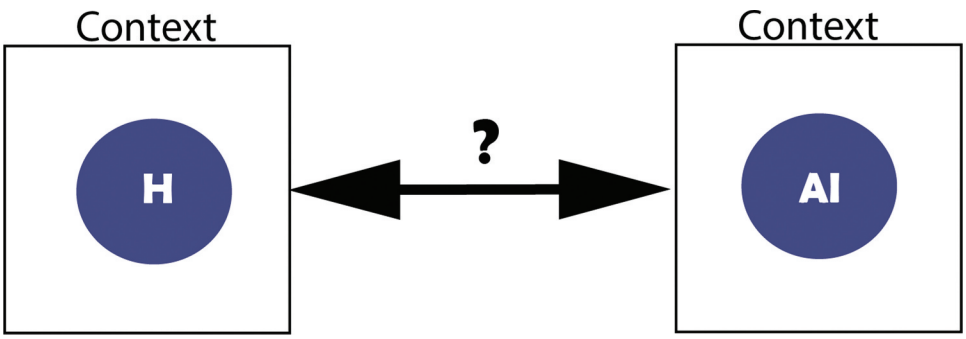

Figure 1. Previous theories examined how humans (H) handling of contexts can be mimicked by Artificial Intelligent systems (AI) underplaying system users and developers.

artificial systems maximize satisfaction across relevant contexts for the humans is intended to serve (see Figure 2). In doing so, we abstract ourselves from the detailed inner workings of the sensorized IE and also of the AI theory that mimics how a human may reason with contexts (as in the KRR multi-contexts area mentioned further up) to concentrate on how to bridge these two areas.

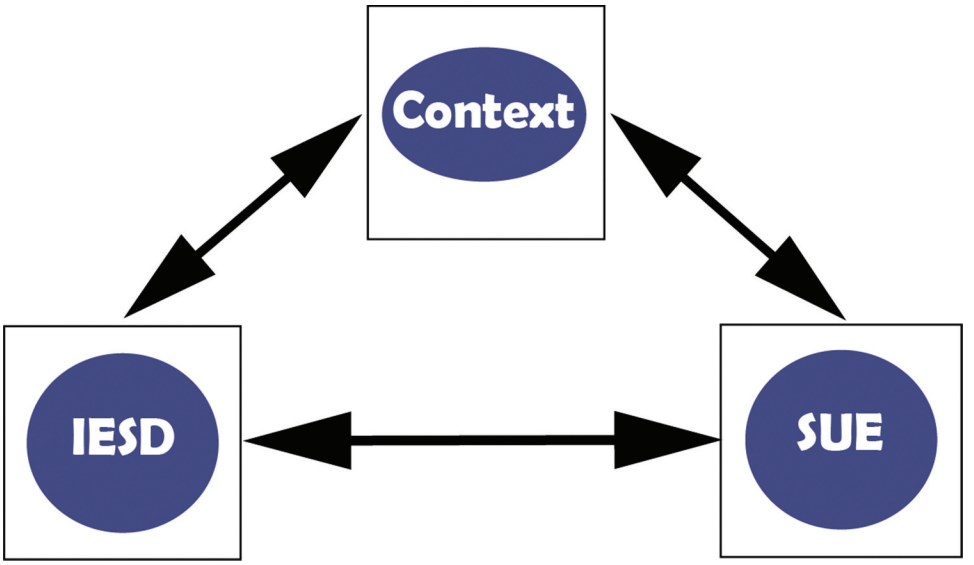

Figure 2. We emphasize the interrelation between contexts, system users experience (SUE) and IE system developers (IESD).

Delimiting Our Analysis

We will now resort to some more technical notations to increase the precision of the discussion, and will keep it as simple as possible to benefit a bigger number of readers and also to avoid diversions into other topics, which although worthy of discussion will significantly increase the length of this article. For example we will assume the system can be represented through some formal system F, for example a First-Order Logic theory, with language L, well-formed formula w, axioms A, and an inference mechanism to create new knowledge from previous one. We will liberally adopt a notion of "logical consequence" represented by the symbol " $\models$ " with the usual understanding by "$A \models a$" that "A" entails "a". We will assume the interpretation of formula into Boolean values. These assumptions are of course important in the final landing of the concepts discussed here into specific systems being implemented, however, the focus of this article is

to discuss the concept of context at a level which is detached from the final implementation. We are trying to clarify first what contexts are, in the extent of interest to the IE-related communities mentioned at the beginning of this article. We will assume mechanisms for consistency checking and knowledge revision, which again we assume adapted to other system choices made at implementation level.

One interesting divergence in the literature is whether each context theory have its own theory associated and possibly different than that of other contexts in that world. Although in the attempt to capture the complexity and richness of the real world it may be tempting to assume each context can have an independent inner system, given our focus on guiding developers of self-contained systems we will focus on systems which are assumed to share the same fundamentals. There are alternatives such as assuming there should be differences amongst components of the system and that these communicate only through the symbols for which they share an understanding of.

Contexts Revisited

In different areas of knowledge the concept of context serve different implicit purposes. In Natural Language Processing it relates more to understanding how it influences meaning. In AI the strongest interest is in how it influences system deductions. In IE historically has been associated with delivering specific services. Initially one of the main contextual dimensions to be used was geolocation, for example, if you were in an unfamiliar part of the city and wanted to find a place to eat, GPS was used to relate your position and offer eating options nearby. As technology increased the diversity of sensors in different gadgets of daily access the range of services offered diversified. In our user-centered approach we differentiate our approach from other more system-oriented and data-oriented approaches by emphasizing the humans that are supposed to benefit with the system:

Context: the information which is directly relevant to characterize a situation of interest to the stakeholders of a system.

Context-awareness: the ability of a system to use contextual information in order to tailor its services so that they are more useful to the stakeholders because they directly relate to their preferences and needs.

Typically in IEs a system is designed so that a group of humans can enjoy the benefits of certain services mediated by technology (Figure 3). From the early stages of system conception, the expectations should be gathered and transferred to the system through design, development, validation, etc. (Augusto et al. 2018). For the system to be of practical utility users should be able (and encouraged) to pass their contextual behavior expectations to the developers and these should find ways to give these contexts contextual relevance and internal mechanisms for the awareness of those contexts.

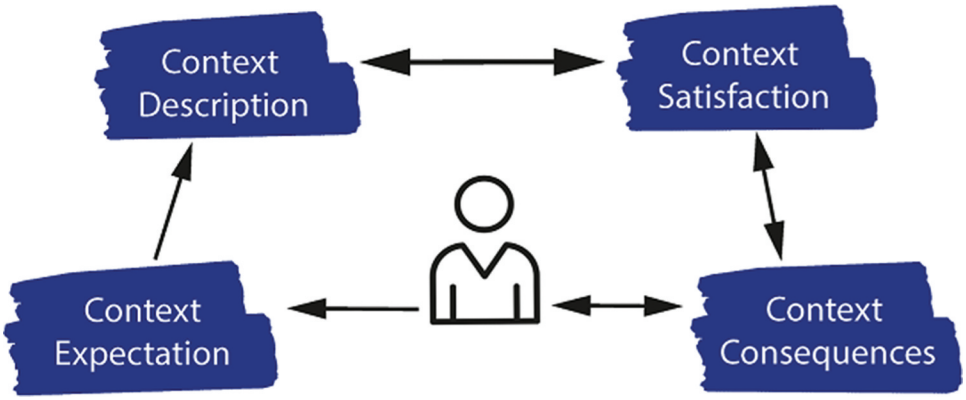

Figure 3. Core context concepts in IEs development and use.

Beneficiaries

We want to highlight the relevance of user expectation and system delivery alignment, which is so relevant for the success of IEs, and we are using this core concept to revisit how contexts are used in this area by shifting the focus from system to human. As a first step in this article we focus on the expectations of one user, leaving for later the next natural step of generalizing to multi-users. Some work on expectations from multiple-users has been recently published (Augusto and Muñoz 2019; Oguego et al. 2018). Meanwhile, here we consciously decide to emphasize in this conceptual revision only the relationship between a single user and the system. Hence, although we recognize it is natural to consider a set of system "beneficiaries" $\mathbb{B} = \{B_1, B_2, \ldots, B_b\}$ we will focus on one of those $B_i \in \mathbb{B}$. Typically in the literature of the area there is reference to the term "stakeholders," also to "(system) users." As this article is written with the intention to help engineering systems in this area it is worth to clarify these terms. The beneficiaries are those stakeholders which are the direct intended recipients of the benefits of the system when it is operative. Developers are also stakeholders, they have a vested interest in the system, and basic infrastructure providers too, however these are not what we will call (main) beneficiaries. There could be other stakeholders such as companies who provide some of the infrastructure such as equipment or networking. Systems in the Ambient Assisted Living (AAL) area also consider Primary users, Secondary users, and Tertiary users (Stefan, Aldea, and Nechifor 2018; Nedopil, Schauber, and Glende 2013). The "beneficiaries" in this article are akin the Primary Users in AAL systems, whilst secondary and tertiary users may depend more on the nature of the system (see Figure 4).

Let us consider some practical situations to supplement the explanation and understanding of our more conceptual discussion. As an example of a possible IE consider a Smart home infrastructure including a combination of sensing/actuation devices and dual function ones. These devices send data to a central

hub so that events can be temporally ordered and processed by algorithms, which implement the automation services (Augusto et al. 2020). Now let us consider a scenario that takes place in such environment:

> *Beneficiary B lives in a smart home and has two main weekly routines from Monday to Friday and during weekends. Monday to Friday routines involve waking up at 7AM to be ready to go to work at 8AM. B expects some automation services. During the process of getting up in the morning B typically gets up from bed, goes to the bathroom, then to the kitchen, has breakfast, and goes out of the house to work. The pressure pad in the bed and motion sensors in the bedroom allows the system to understand when B is physically getting up and other motion sensors in the corridor, the bathroom and the kitchen allows tracking B's trajectory. The system turn on lights in the next relevant room and turns them off in rooms where they are perceived not to be useful anymore. When the user enters the kitchen the system turns on the radio. The sensors in the doors of the kitchen coverts and fridge as well as the devices used, such as the kettle or the microwave oven, provide clues of the user preparation of breakfast. Meanwhile the system can present information on weather and air quality air for the work area of the city, which helps B's decision making about transport choices to reach the workplace. It could also happen that B gets up during the night to go to the bathroom and the system is expected to understand this is not the same than the breakfast routine to go to work. One of B's elderly parents, P_B, also lives in the same house and has been increasingly experiencing symptoms of senile dementia, with increased safety risks, so the system is expected to differentiate between different beneficiaries going to the bathroom and trying to go out of the home and at what times these events are expected. Guiding lights can reduce P_B's risks of falling and also help with her orientation. Getting up in the middle of the night for the bathroom should not trigger actions in the kitchen. B's leaving going out of the house during dark hours is fine but it may be dangerous for P_B given the spatial and temporal confusion experienced for that person.*

The scenarios above may look simplistic at first glance; however, detecting these types of contexts and correct decision-making is at the core of intelligent environments expectations and appropriate context detection and context trigger are actually loaded with ambiguities, contradictions, priorities and other interesting challenges. Unsurprisingly, the getting up morning context is so important in our daily life experience that it was part of the several systems we developed. For example, to provide personalization in determining alignment with a healthy lifestyle for users experiencing the onset of dementia, in providing an environmental assessment to an asthma sufferer before heading for work, and in making the bedroom to provide lighting services according to complex dynamic preferences (for more details, see (Augusto et al. 2020)).

Services

We have brought to the forefront of our analysis that IE systems are created with the intention to benefit humans in specific circumstances. The environments can be very diverse, it could be our home, our workplace, our car,

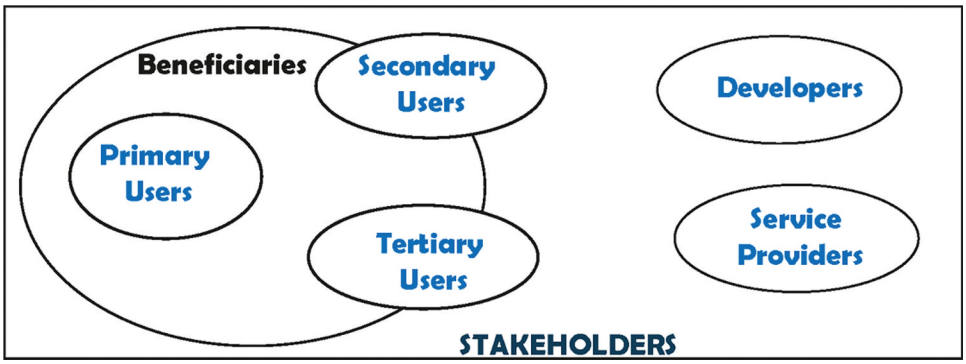

Figure 4. Distinguishing stakeholders and beneficiaries.

a plane, a shopping center, an airport, a hospital, a manufacturing plant, a street in the place where we live, a museum or art center. All of these places can be enriched with technology and intelligent software to serve humans through context-awareness. Benefits can also be very diverse, and could go from leisure time and comfort, to more important aspects of health, safety and security. Some of these systems can be dedicated to increase efficiency and safety at work, or to provide information and guidance. The way these benefits are achieved in a context-aware manner can be very diverse too. Sometimes adjusting environmental parameters such as light and temperature, or delivering recommendations through a screen or in spoken mode through a device. In machines which we depend on the benefit is achieved through the regulation of the machine itself (e.g., the car we drive or the plane we are in) by adjusting certain parameters which are directly related to our objective of achieving a certain goal in a certain manner (e.g., adjusting steering in icy road conditions or assisting the landing of a plane in foggy conditions). We will assume in any IE there will be a finite number of "services" $\mathbb{S} = \{S_1, S_2, \ldots, S_s\}$, which can be represented and organized in various different ways. There could be a complex hierarchy of them, ontologies, etc. Being exhaustive or prescriptive on this element of IEs is beyond the scope of this article and we will describe them only in a superficial and intuitively to the extent it helps completing the picture of how contexts work.

Contexts

Context is a concept which we humans use to understand the world and to function within it in best way we can given the partial knowledge we have about it at a given time. In different areas of knowledge "context" is interpreted within a specific set of assumptions and conventions of that area. In Linguistics it is related to how language constructions change meaning, in AI how

inferences are affected when certain context-dependent assumptions are made. Here we will take that in IEs these properties should relate to beneficiary's expectations of the system. We can consider a context as a state of the world with context-specific priorities, in our case we are most interested in these in relation to the beneficiaries of the system. Say the system representation of the (fragmented and narrow view of the) state the world is in, WS_j, consist of all the truths in that system at a specific "$j-th$" stage (we address how this relates to time further down). There are also meaningful terms which have been used for decades, one of those evoked is that of "Situation," proposed by AI pioneer John McCarthy and largely studied as the Theory of Situational Calculus. McCarthy explicitly stated (McCarthy 1990) that contexts are not situations, that situations happen within a context, and that by the context changing, the inferences over situations change in outcome. Situation Calculus still have that AI flavor of being centered on the system inferences whilst Context-Awareness as we are exploring it in here is based on the user subjective experience of services in specific contexts. Some may be tempted to "patch" previous theories, however we will miss the chance to rethink CS, AI, and IEs afresh, which will start to manifest in sections below. Let us consider a finite set of contexts $\mathbb{C} = \{C_1, C_2, \ldots, C_c\}$. We will assume each context $C_i \in \mathbb{C}$ will have a set of Activating Conditions AC_i which need to be satisfied for the context to be "active." AC_i conditions are expressed in some language L and is such that $AC_i| \neq \perp$ (is not internally inconsistent). At design level they may be described by requirements, for example, that context "night time" will be active when clock time is between 9 PM and 6 AM, or when is dark outside). At theoretical level this can be represented in various ways, in a symbolic approach for example the theory can be based in a Logical language with formula "within(9PM, T, 6AM) $\rightarrow$ night_time(T)" and "lumens_outdoor(Lux, T) $\wedge$ (Lux $<$ 1) $\rightarrow$ night_time(T)." Notice although for simplicity we resort to a symbolic Boolean style representation, however different approaches (Fuzzy, Connectionist, etc.) are also feasible. At implementation level they will be represented through code in some implementation language as Java, Python, or C#, which will check values of the system clock and luminosity level sensors. In addition, our approach is very much "syntactic" in style; however, a more semantical emphasis (what is implied by AC_i) can also be done. Of course each option will have specific expressiveness and computational complexity pros and cons associated. We are not prescribing over these options, our discussion is at a conceptual and complementary level. In singling out some contexts we will adopt a simple structure to highlight different components which will be useful for presentations later on. So far there has not been a deep discussion in this community on how a context can be represented. Let us call them "Context Templates". We propose that at least an essential template can include: *[Name, Beneficiary, Activation, Effect]*. Where *Name* is a string that univocally represents the context within the system, *Beneficiary*

could be an individual or a group of them, *Activating Condition* can be composite and include all the circumstances where the context is supposed to be active, and *Effect* encapsulates the services which are triggered by the context. Doing nothing could be a service too if in line with keeping the beneficiary pleased. Some examples of contexts templates representing contexts from the scenario described before are listed in Table 1.

Contexts will be identified by their name, we will adopt a Unique Names Assumption on the name, and the context name will reflect the context the developer had in mind. Each concept can be then manifested in different meaningful ways, as there are for example, more than one way that night time can be defined and some contexts can be linked to more than one person. So we will consider the more abstract context concept and the different context instantiations. Above we provided five context concepts and for each of them there are two instantiations, one for each of the two beneficiaries mentioned in the scenario. Context instantiations can be differentiated by the name and also compared by any of the other fundamental aspects: Beneficiary, Activation, Effect. Organizational and optimization issues at implementation level are not the subject of this article, support for using the concepts here discussed in an optimal way is a matter of future publications.

Table 1. Examples of context descriptions based on the smart home scenario.

	Context concept: front door use	
Name	Front_door_use1	Front_door_use2
Beneficiary	B	PB
Activation	Any time ∧ @ Front door ∧ Open door ∧ Leave house	8 AM–8 PM ∧ @ Front door ∧ Open door ∧ Leave house
Effect	Do nothing	Inform B
	Context concept: Going to bathroom	
Name	Going_to_bathroom1	Going_to_bathroom2
Beneficiary	B	PB
Activation	7:00–7:15 ∧ Gets up from bed ∧ Bedroom movement	7:00–9:00 ∧ Gets up from bed ∧ Bedroom movement
Effect	Turn on lights in bedroom, corridor, and bathroom	Turn on lights in bedroom, corridor, and bathroom
	Context concept: Getting up routine in process	
Name	Getting_up_AM_routine1	Getting_up_AM_routine2
Beneficiary	B	PB
Activation	7:00–7:15 ∧ Enter bathroom	7:00–9:00 ∧ Enter bathroom
Effect	Turn on radio and kettle	Notify B
	Context concept: Skipping lunch	
Name	Skipping_lunch1	Skipping_lunch2
Beneficiary	B	PB
Activation	12:00–2:00	12:00–2:00
Effect	Do nothing	Remind PB
	Context concept: Being at kitchen	
Name	Being_at_Kitchen1	Being_at_Kitchen2
Beneficiary	B	PB
Activation	Kitchen PIR activated	Kitchen PIR activated
Effect	Log activity	Log activity

The Observable World

Each IE happen in a physical part of the world, the system and beneficiary have limited perceptions and experiences of that part of the world. IE perceive the world through the information they gather from technology. We will call that the Observable World, $\mathbb{OW}$. We assume the representation of it in the system will be based in either the same language used for the Contexts Description or that at least there are ways to interpret one into the other. This linguistic bridge is important to enable different essential stages of the context usage: to understand the context the world is in and to influence the environment (i.e., the observable world). There is a wide gap in this area between the higher level language of contexts description (e.g., "someone is in the kitchen") and the way these contexts are detected through technology (a PIR sensor in the kitchen has been triggered, i,e., $PIR_{kitchen} = ON$). Let us assume the formula higher-level context language is F_c and the lower level language of sensing in the observable world is F_o (and that these two languages are comparable). Then we assume there will be processes in the system to check whether a context condition described using well-formed formula (WFF) w_c in F_c is fulfilled by the meaning of WFF w_o in F_o. Let us name this condition fulfillment check: $f(w_c, F_c, w_o, F_o)$. Which checks the semantic satisfiability of (possible composites) w_c and w_o. The Observable World related to the IE system changes as time passes and events occur which have an effect in it. These real-world phenomena will be typically stored as logs of timestamped events in databases, which can be then analyzed by typical reasoning and machine learning algorithms to check for conditions or find patterns of interest. We can assume a structure $\mathbb{OW} = \{\ldots, OW_i, \ldots\}$ which we assume here to correspond to a discrete and linear time view, where OW_{now} is the current observable world and we can use OW_t and OW_{t+1} as notation to indicate two consecutive slices or snapshots of that observable world. Various modeling choices of $\mathbb{OW}$ and its dynamics can be considered by developers. For example, if a Newtonian model of change is assumed then time passes relentless and it may be that in between two consecutive measures of the time granularity of choice (e.g., seconds or minutes) there was no perceived change. Whilst in a Leibnitian approach we may associate this observable world slices to events, noticeably including events, which may have no practically useful effect (Augusto 2001).

Intelligent Software

The task to deliver services which are aligned to the beneficiary expectations can be a very challenging one even in the simplest of scenarios. It is actually quite interesting to see the different dimensions influencing decision making in services automation, which in the surface looks deceitfully simple. The service in itself is only one aspect, there is also the delivery mode: When? Where? Which

channel this information should be conveyed through? Which communication mode should be used? How immediate? For how long? The system intervention in the environment and its interaction with the beneficiary can be influenced by a complex combination of information. Sometimes the information is gathered and created in real time and in a reactive mode. Sometimes the decision making is informed by data gathered through days, weeks or months to create a profile of preferences, habits, and needs. In some cases, deductive reasoning algorithms are used, other times machine learning style algorithms are more useful. Systems with a complex range of concepts and interrelations can require an ontology. All these areas of computer science need to be combined to achieve satisfactory higher level automation. Again, typical of this area, there are not standards used by a majority and often systems are created ad-hoc with little in the way of transferability, the SEArch architecture (Augusto et al. 2020) is one attempt to start a discussion on the different algorithms, which may form part of the usual toolkit for systems in this area. Here we do not rely on specific algorithms; however, we recognize the undeniable importance for a theory in this area to use them and we will indicate them in a generic way. We will consider a number of information processing "algorithms" $\mathbb{A} = \{Al_1, Al_2, \ldots, Al_a\}$ which each team can instantiate with their favorite problem-solving strategies (e.g., fuzzy, spatio-temporal, ANNs, Markov models, ontologies, etc.).

A Theory of Contexts for Intelligent Environments (CIEn)

Our Contexts for IEs theory (CIEn) is a structure focusing on a set of contexts and a set of operators which can help with reasoning over those contexts: $\mathbb{CIEm}=\langle \mathbb{B}, \mathbb{S}, \mathbb{C}, \mathbb{Ops}, \mathbb{A}, \mathbb{OW} \rangle$ where

$\mathbb{B}$ is a finite set of beneficiaries,

$\mathbb{S}$ is a finite set of services,

$\mathbb{C}$ is a finite number of contexts $C = [Name, Beneficiary, Activation, Effect]$,

$\mathbb{Ops}=\{\text{Op1,Op2,}\ldots\}$ is a finite set of context operations (see next sections),

$\mathbb{A}$ is a finite set of algorithms,

$\mathbb{OW}$ is a finite set of instances OW_i

and various conditions hold such as: $OW_i \subset \wp(L), OW_i \not\models \bot, AC_i \subset \wp(L), AC_i \not\models \bot$. There are of course several open aspects of the theory we cannot cover and can be addressed in future work. Each of these components of the theory can have more specific theories on their own. Toward the end of this article we retake the holistic aspect of the theory. Earlier sections looked at contexts from a conceptual level and the importance of considering beneficiaries explicitly as key actors in the motivations for contexts being considered in a system and also as the ultimate measure of success for the context-awareness of the system. Now we are delving into the inner side of the system, the key steps and conceptual tools required for context creation, and in doing so we turn our attention more to the needs of developers. Contexts are

considered from different perspectives during the development of an IE. Whilst this process can be analyzed at different levels of detail here this article considers three main stages:

(1) Context Definition encompasses all the initial conception stages, from understanding there is a need for a specific context to be considered in the system, to define it conceptually and to design it.
(2) Context Use captures all what is the context deployed in the real world, including detecting that the context is actually happening and triggering the associated actions.
(3) Context Development refers to all steps of materializing the context in the system as part of its functionality and this will include tasks as programming and testing.

Figure 5 captures these and bidirectional arrows are used to represent each of these stages can influence each other. The team has to go through a number of iterations until they converge into a context which matches the expectations at all levels, conceptual and practical, for all stakeholders.

Context design: Turner (Turner 1993, 1998) introduced Context-mediated behavior (CMB) and C-Schemas to manage context-related concepts and processes in relation to an autonomous artificial agent. Inspired in the minimalistic approach suggested in that work we further elaborate those thoughts into a sort of "Occam-Razor rule for context design" where: A context should be implemented in the system if and only if:

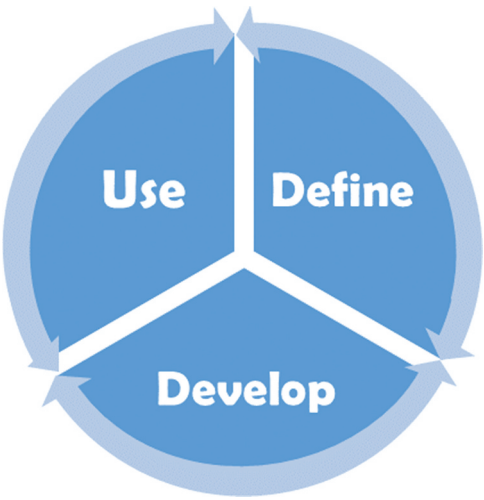

Figure 5. Basic cycle of contexts life-cycle in IE development.

(1) It is necessary (i.e., it contributes positively to the Beneficiary Context Expectations, or it supports a context that does that),
(2) It is not currently in the system (directly or by inference from others), and
(3) It can be implemented (within the resources of the system).

The definitions above assume contexts interrelate to each other and that is a reasonable assumption, IE systems usually consist of more than just one context and getting the system right requires orchestrating those contexts so that they collectively provide beneficiaries with the services they expect. If *B* enters a room and is lit then there is no need for turning on lights but when dark the system is expected to turn on the lights. However, if it is 3 AM and the user is going to the bathroom then lights should be dimmed instead of fully bright. Trying to get an IE to be more practically intelligent naturally leads to several variations of interrelated contexts. Now, how can these contexts interconnect? Again, without presuming to be exhaustive we explore a couple of directions.

Context Definition has been addressed at various levels in sections above. The sections below focus on Context Use and Context Development in IEs.

Context Use Operations

Given a set of contexts C as discussed in the previous section and a mechanism to track the observable world OW, we first consider here some operations in Ops which can be used by developers to reason about the contexts they need to create.

Context Satisfaction

Contexts are usually defined in an IE working system because there is an expectation the system will do something different in that context. For this process to take place the system needs to check first whether a context is satisfied or not, then if it is satisfied it needs to trigger the consequences. Given a context $C_i \in \mathbb{C}$ defined by $AC_i = \{w^1_{c_i}, \ldots, w^n_{c_i}\}$ and given $\mathbb{OW} = \{\ldots, OW_j, \ldots\}$ such that $OW_j = \{w^1_j, \ldots, w^m_j\}$, we can define a context satisfaction function $\surd : C \times OW \rightarrow Bool$, which decides whether the observable world of the system satisfy the context or not. This operator will basically check whether $OW_j \models AC_i$ and it can be defined as:

$\surd(C_i, OW_j) =$ "true"

iff forall $w^x_{c_i} \in AC_i$ there exists : $w^y_j \in OW_j$ such that $f(w^x_{c_i}, F_c, w^y_j, F_o)$

and false otherwise

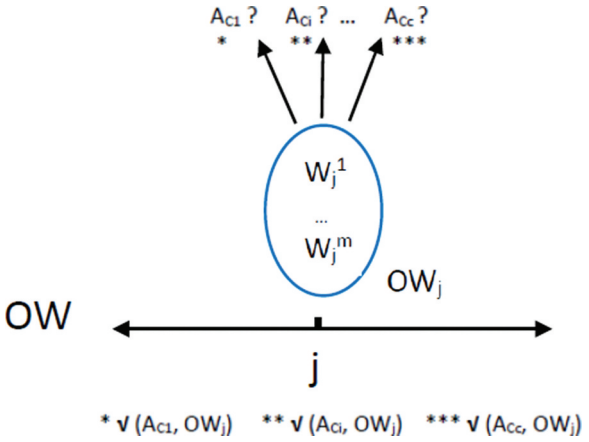

Figure 6. Diagrammatic depiction of context description and actual contexts relations.

Figure 6 shows a generic depiction of how these concepts relate to each other in practice. Typically OW_j bigger than AC_i (in a subset-inclusion sense) and C_i more abstract than OW_j (as the former contains generic conditions and the latter specific data). An index "j" can typically be interpreted as "now," the present, however it could also refer to a hypothetical future time or to one in the system history.

Some context satisfiability checks can be instantaneous whilst some will be durational. For example the context $C_{LateForMeeting}$ can be triggered if B is still at home at 8:10 AM whilst the context and all the information we need to check is the one we have at that time. Whilst $C_{LeavingCookerUnattended}$ defined as: "nobody has been in the kitchen in the last 10′ whilst cooker is on," can only be triggered after the 10 consecutive minutes of the condition being uninterruptedly true. More on Durative Events detection in (Galton and Augusto 2002). We do not delve in this direction here and we assume instantaneous checks from which durational ones can be defined (Augusto 2003). Also given satisfiability provides a Boolean outcome, other classic Boolean connectives can be used, and we can express when one context is not active: $NOT\surd(C_i, OW_i)$, or when several contexts are simultaneously active: $\surd(C_1, OW_{t_1})AND \ldots AND\surd(C_n, OW_{t_n})$, or when at least one of a few is active: $\surd(C_1, OW_{t_1})OR \ldots OR\surd(C_n, OW_{t_n})$. Of course these can be combined in the usual matter and there is a parallel here with composite (or "complex") event detection (concepts steaming from Active Databases community), see for example (Galton and Augusto 2002).

Contexts in Relation to Other Contexts

Different functions $\Psi([Name, Beneficiary, Condition, Effect], OW_j)$ can be defined to characterize and differentiate contexts based on their internal conditions and how they relate to the world they interact with. This can be generalized

to conditions which can be stated at a higher level and affect more than one context in relation to some world observations: $\Psi(C_1, \ldots, C_n, OW_{t_1}, \ldots, OW_{t_m})$. For example say we want to create a context out of two smaller contexts to state that context "B is out of home" being detected at the same time than context "movement detected at home" leads to the context "potential home intruder," in which we want to notify B of that. This brings us back to the role of time in context detection. We need to pause a bit to clarify some notation, previously we mentioned context detection associated with time points and we also mentioned an example of context detection which is durational. A time point based conception of time can be used to define a durative conception of time (Augusto 2003), this can be used for durative detections of contexts. A succession $OW_{t_1}, \ldots, OW_{t_n}$ can be compactly referred as $OW_{[t_1,t_n]}$ to refer to the observable world during an interval $[t_1, t_n]$ as in 9 AM–11 AM, night time, weekday, etc. Then more sophisticated Ψ conditions can be stated about how a group of contexts can or should coexist temporally in order to detect that a higher-order event has occurred. For example:

$$\begin{aligned}\surd(C_{intruder}, OW_t) = def \\ &\surd(C_{B\_is\_out\_of\_home}, OW_{interval}) AND \\ &\surd(C_{movement\_detected\_at\_home}, OW_t) AND \\ &(\Psi(C_{B\_is\_out\_of\_home}, C_{movement\_detected\_at\_home}, OW_t, OW_{interval}) \\ &= during(t, interval))\end{aligned}$$

where here $during(time, interval)$ can be interpreted as in Allen's-Hamblin's theory of time intervals (Allen 1983; Hamblin 1972) and the AND operator can be assumed as in propositional logic.

Context Comparison

Within the context-development stage developers have to consider each context in relation to other system contexts, as rarely there is only one context in a system and contexts interact with each other (even if implicitly as complementary). An essential part of this process involves comparing contexts. This comparison can be assessed at various levels of detail, so we only offer here a first glance at the possibilities opening up in this direction. Let us assume two contexts C_1 and C_2, they can be contrasted with each other through a specific aspect such as the activating conditions, or their beneficiaries, their effects, or any combination of those, or any other context aspects designers wish to include and highlight in the contexts as they conceive them in the system. Therefore some generic operations will have to be defined by developers on these context dimensions (e.g., beneficiaries, conditions, effects), also based on the comparison of specific dimensions other higher order comparisons can be defined. We can retrieve the contexts which relate to a specific beneficiary or set of beneficiaries, or those with certain

effects or with certain characteristic features (e.g., temperature related). This comparison will only make sense on what we called context instantiations further up as the more abstract context concept will be unique. For example, given two contexts C_1 and C_2 defined by the four aspects given early on: *[Name, Beneficiary, Activation, Effect]*, each one generically referred to as A_i, we can consider:

"C_1 is equal to C_2 on aspect A_i", represented by $\ominus(C_1,C_2,A_i)$ when aspect A_i in both C_1 and C_2 have the same content; for example contexts which are of interest when they manifest in the same place or at the same time, or relate to the same beneficiary.

"C_1 overlaps with C_2 on aspect A_i", represented by $\otimes\ (C_1, C_2, A_i)$ when aspect A_i in C_1 and C_2 have some common content; for example a context which share some specific times of the day when they coincide or coincide some of the services used, e.g., an alert message sent, but not other services which are unique to each of the contexts.

"C_1 subsumes C_2 on aspect A_i", represented by $\oslash\ (C_1, C_2, A_i)$ when aspect A_i in C_1 contains all elements of the same Aspect in C_2 and more; for example, when comparing contexts which apply to all members of the family and to a subset of them (e.g., kids only).

Of course there is also a parallel between the operations above and the all too familiar set-theoretic ones and it is clear there are more economic ways to present these from a purist point of view with a fundamental operation and other operations being derived as a consequence (e.g., equals when there is overlap and neither subsumes each other). There is also the point on whether comparison should be syntactic or could it be more semantic in nature (equals and equivalent), which is not at the level of details we are exploring the subject in this article.

Typically two contexts which are not in any of the above relations, that is, there is no relevant aspect of comparison where they connect including the conditions in which they happen, can be perceived as unrelated. Developers can still organize contexts 'ad libitum,' into more rigid or flexible clusters which help their conception of the system, including an organization based on meta-system properties which are not expressed in the context templates themselves. For example, safety of individuals may not be an explicit part of the context template and still developers consider it important in the system and use it as one dimension from which to conceive the system. Applying these to the scenario introduced further up in this article and based on the context descriptions listed in Table 1, we have:

$$\ominus(C_{Being\_at\_kitchen1}, C_{Being\_at\_kitchen2}, Activation)$$
$$\otimes(C_{Skipping\_lunch1}, C_{Skipping\_lunch2}, Activation)$$
$$\oslash\ (C_{Getting\_up\_AM\_routine2}, C_{Getting\_up\_AM\_routine1}, Activation)$$

Similar statements can be made on other aspects. In this reduced scenario, all comparisons of different instantiations will of course indicate different beneficiaries, however it is varied for they are different in the effects. However the most interesting case is brought by the activation condition of the first context concept Front Door Use, as if the comparison is syntactic they can be said to be overlapping: $\otimes(C_{Front\_door\_use1}, C_{Front\_door\_use2}, Activation)$ whilst a more semantic comparison will interpret that the time range in the second is included in the most generic one and therefore: $\oslash\ (C_{Front\_door\_use1}, C_{Front\_door\_use2}, Activation)$.

Context Influencing Other Contexts

After understanding what contexts there are in the system and how they are triggered, one other essential view of system contexts developers need to understand and craft is how these contexts influence each other.

"C_1 *Directly Influence* C_2", $C_1 \triangleright C_2$: if we have $C_1 = [1, *, *, Effect]$ and $C_2 = [2, AC_2, *, *]$ then we can say C_1 directly influences C_2 when the effect of a context C_1 has a direct impact on another context C_2, that is, if we consider $C_1(Effect)$ the symbolic representation of the services triggered then $C_1(Effect) \models S$ where $S \subseteq AC_2$. For example, $C_{B\_GetsUp} \triangleright C_{PB\_GetsUp}$ could be an inferred context if the machine learning algorithm correlates historic data for *B* getting up resulting on *PB* getting up as well. Notice this should not be interpreted as "causality." Context Inheritance is a special case of Direct Influence, for example, if we assume "homogeneity" of properties, a hot house means each room is also hot. That is, there are context influences that can occur more naturally (e.g., through inheritance) and can be set to happen in a more automated way through rules in the system, whilst in other cases they will have to be designed in a more ad hoc way by the developers.

"C_1 has a *Ripple Effect* on C_2", represented as $C_1 \triangleright\triangleright C_2$: this can be seen as Indirect Influence, the most complex and most interesting case, where events occurring within one context, C_1, affect another context, C_2, but in a less obvious way. This is actually a common phenomenon in the world. When there is a car crash in the motorway it affect the mood of many drivers, when an airline goes bust it affects emotionally and financially so many people, including customers and employees. Market oscillations, winning the lottery, epidemic outbreaks. We are always affected in small and big scale by what happens around us. There is an infinite number of continuously interacting micro and macro contexts. The Ripple Effect of contexts could be a difficult concept to implement or predict, however IE system design should discuss it and reflect it even if in a watered-down version. It has big practical value given it is frequent and a powerful

influence in real life situations and mastering this concept will allow to create more subtlely intelligent environments. Consequences of Ripple Effects can be positive or negative. If some context C_i have known negative consequences associated then developers can create a number of context detection mechanism, which helps identifying when the circumstances are becoming favorable for the satisfaction of AC_i and then the system can then take preventive or remedial actions accordingly. In the context of winter time and a house with central heating, the lowering of the temperature to satisfy the comfort preference of the residents may be prevented if it clashes with the health need of another resident. $C_1 \triangleright\triangleright C_2$, can be characterized as follows:

1. C_1 has to start at least no later than C_2 finished
2. there are C_1-related events which occur during the span of C_1 and affect the value of properties in AC_2

Condition (1) can be addressed by checking satisfaction of instant-based time relationships, for example: begin(time_span(C_1)) " $\prec$ " end (time_span (C_2)) (Augusto 2003). Notice that a definition based on Hamblin's and Allen's style intervals (Allen 1983; Hamblin 1972) gets cumbersome. Condition (2) can be described as the satisfaction of: $\surd(C_1, OW_t) \models C_1(Effect) \models \ldots \models C_2$.

Contexts Organization

How contexts can be organized? How contexts interrelate and how that informs and affects the way developers approach a system? There is little in the literature about these fundamental questions. We cannot answer all these questions in a decisive way, this should be the subject of a wider scientific community discussion; however, these questions set the scene for the concerns driving this part of the study. Ontologies could be considered as a way of organizing a concept and there have been attempts in this direction. There were some theories (Bouquet et al. 2003b; Obrst and Nichols 2005) and attempts at reusing existing tools to help creating context categories from smaller system data (Chen, Finin, and Joshi 2003; Gu et al. 2004) proposed a general purpose context model used for pervasive applications, named SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. Whilst ontologies for Ambient Intelligence systems have been proposed, including GAIA (Ranganathan et al. 2003), CoDAMoS (Preuveneers and Berbers 2005), OntoAMI (Santofimia et al. 2008), and BOnSAI (Stavropoulos et al. 2012). However, no much consensus has been achieved amongst these independent efforts. They are mostly focused either at the level of infrastructure, services or specific application area instead of at context (as a concept) level. Whilst these initiatives have potential and show specific faces of a polyhedron concept, more thinking is required at a more generic and holistic level. So, without ruling out ontologies as a way to be explored further

in the future, which other alternative conceptual tools can we provide for developers to think about contexts in a way which may guide how they approach system development? Below we consider some alternative and complementary ways of looking at this, starting with the genesis of contexts.

Programming-constructor Style

There is some resemblance between contexts and program units which have local properties at the same time they communicate with other external units and are subject to higher order properties which somehow conditions them. In developing context-aware systems we often find ourselves working with the very familiar algorithmic building blocks:

Sequenced processes: $\surd(C_1, OW_{t_1})$ *FOLLOWEDBY* $\surd(C_2, OW_{t_2})$ *PROVIDED* $\Psi(C_1, C_2, OW_{t_1}, OW_{t_2})$ For example, *B*'s movement in the bedroom following by user movement in the corridor can define the context of transitioning from bedroom to corridor. The condition imposed by the developers could be here that they happen "temporally close enough," e.g., t_1 and t_2 are consecutive instants or that if they are intervals that they either "meet" or "overlap" (in interval-based relationship terms).

Conditioned processes: $IF \surd(C_1, OW_{t_1})$ *THEN* $\surd(C_2, OW_{t_2})$ *PROVIDED* $\Psi(C_1, C_2, OW_{t_1}, OW_{t_2})$ For example, *B*'s getting up from bed in the middle of the night will be a requirement to turn on a (dimmed) bedroom light. Entering the kitchen in the morning will be a prerequisite to turn on the radio. The condition imposed by the developers could be here that $t_1 = \ldots = t_n$ and that $t_z = t_{n+1}$.

Cyclic processes: $LOOP \surd(C, OW)$ *PROVIDED* $\Psi(C, OW)$. For example, going so many times to the bathroom during the night being an indicator of a health issue. The condition imposed by the developers here could be one of $Frequency(t_1, \ldots, t_n)$ that is the detection happen certain number of times during the night, perceived to be the "too often" threshold. Notice the "conditions" of a 'conditional type' of clause is made up of other contexts being triggered. Whilst the " ... *PROVIDED* $\Psi(C_1, \ldots, C_n, OW[t_1, \ldots, t_z])$" part of each of the three constructs above relates composite contexts to higher level circumstances, we can call them "meta-contexts." In addition, we assume composition of contexts is possible, that is when a context *C* is mentioned in any of the operators above it could be a "simple/atomic" event or it could be a "complex/composite" one. This assumes context triggering can be distributed, so that $\surd(C_1\ OP\ C_2, OW)$ can be decomposed into $\surd(C_1, OW)\ OP\ \surd(C_2, OW)$.

For example, in our scenario we can represent the following complex contexts:

$$
\begin{aligned}
C = [&GoingToWork, B, TimeLeftHome, OutsideHome]=_{def}\\
&[[\surd([GoneToBathroom, B, BTime, atBathroom], OW)\\
&FollowedBy\surd([GoneToKitchen, B, KTime, atKitchen], OW)\\
&]FollowedBy\surd([FrontDoorUse, B, FTime, outsideHome], OW)\\
&PROVIDED(weekday(OW)ANDbetween(OW, 7-8AM))\\
&]
\end{aligned}
$$

which states that going to workplace routine typically involves B going to the bathroom, then to the Kitchen, followed by exiting the house. More constraints can be placed on the duration of these activities through the more specific components or other aspects of the context.

$$
\begin{aligned}
C = [&ContactForEmergency, B, TimeC, contacted]=_{def}\\
&LOOP\{\surd([SendMessage, P_B, TimeM, messagesent], OW)\}\\
&PROVIDED(emergency(PB, OW)\ AND\ noAnswer(B, OW))
\end{aligned}
$$

to indicate in an emergency of P_B a message should be attempted to reach B whilst B does not acknowledge reception.

$$
\begin{aligned}
C = [&MealReminder, P_B, TimeR, reminded]=_{def}\\
&IF\surd([SkippedMeal, P_B, TimeS, noMeal], OW)\ THEN\\
&\surd([ReminderSent, P_B, TimeR, message], OW)\\
&PROVIDED\ after(TimeR, TimeS)
\end{aligned}
$$

so that a specific message submission is conditioned by a specific emergency.

Interaction Modalities

One other way to look at contexts with practical utility is the nature of their interactions. Some examples follow which are of practical usefulness when we develop context-aware systems. Let us assume $C_1, C_2, C_3 \in C$ but C_1 and C_2 are at the same level of complexity or abstraction whilst C_3 is a "supra" context of higher relevance to which C_1 and C_2 contribute.

Neutral: $C_1 \circledast C_2$, contexts do not influence each other, given any $p_1 \in AC_1$ and $p_2 \in AC_2$ then $AC_1 \not\models p_2$ and $AC_2 \not\models p_1$, for example, "$B1$ is getting up" and "$B2$ is in bed".

Cooperative: $C_1 \oplus C_2$, both contexts can be combined to realize another one C_3, that is given any $p_1 \in AC_1, p_2 \in AC_2, p_3 \in AC_3$ and $\{p_1, p_2\} \models p_3$, for example, "B wakes up", "B moves" to achieve "lit up room."

Competitive: $C_1 \ominus C_2$ means when either is detected the other one is not, they "turn off" each other by disabling some conditions in the context description, $p_1 \in AC_1, p_2 \in AC_2$ and $\{p_1, \ldots\} \models \neg p_2$, for example, the house pondering between turning on the lights for B's comfort, or not turning on the lights for B's economy.

Incompatible: $C_1 \otimes C_2$ means they cannot coexist simultaneously at any time $p \in AC_1$ and $\neg p \in AC_2$, for example, B being or not being at home.
Examples based on our scenario can be:

$$[gettingUp,\ B1,\text{–},\text{–}] \circledast [staysInBed;B2,\text{–},\text{–}]$$
$$[gettingUp,\ B1,\{p1,\ldots\},\ precond1ToLit] \oplus [bedroomMove,B1,\ \{p2,\ldots\},\ precond2ToLit]$$
$$[luxSitInSofa,\ B1,\ \{turnLight1\ On,\ldots\},\ strongLight] \ominus [ecoSitInSofa,B1,\ \{turnLight1Off,\ldots\},\ weakLight]$$
$$[atWork,\ B1,\ \{atWorkPlace = true,\ldots\},\text{–}] \otimes [atHomeKitchen,\ B1,\ \{atKitchen = true,\ \ldots\},\text{–}]$$

This intercontextual relationships can be implemented at different levels, conceptual as above, or as instantiations with reference to specific conditions of the OW where these are supposed to be valid. One way of achieving the later could be by combining these relationships with the Provided clause introduced further up.

Going back to the definition of our $\mathbb{CIEm}$, where we firstly explained the ingredients of the system and in the last few sections we have given examples of operations which can be used for system processing, so as an example of the system operations and to complete the $\mathbb{CIEm}$ definition, in this article we considered a number of operators in different categories: $\mathbb{Ops}$ = {Activation, Comparison, Influence, Construction, Interaction}.

CIEn Holistic View

The sections above presented the components of a theory which describes an IE with emphasizes contexts, its elements and operations which can be used to work with those elements. Most consideration were "local" in nature, at a micro-context level, including the role of beneficiaries. However beneficiaries can have also a macro-contexts level, a more holistic driving on how an IE behave and how its performance can be assessed. We start by highlighting some additional beneficiary related concepts (Figure 7): *Beneficiary Context Perception (BCP)* is the context as perceived by the beneficiary, where Perception here is understood as measured with the available infrastructure. *Beneficiary Context Expectations (BCE)* are the services the beneficiary expects in a given context.

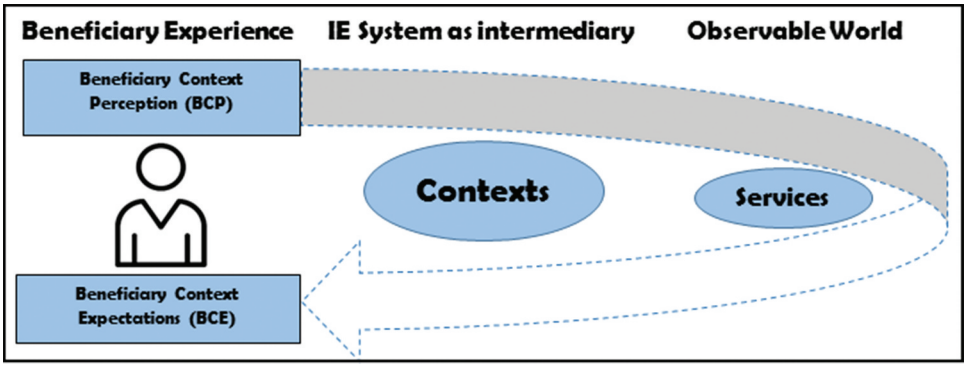

Figure 7. Close up of beneficiary relation with context.

An example could be the light was automatically turned on when the beneficiary entered the room but it took too long, it was expected to be on in 2 seconds and it took 5 seconds. This gap between "system artificial perception" and "beneficiary's natural perception" is the most interesting challenge to developers in IEs. Take temperature, the system may perceive a bedroom as adequately temperate however the beneficiary under a bed quilt may be significantly hotter. Take noise, a place may be noisy however if the beneficiary has earplugs on then may not perceive the context as noisy. Preferences and expectations have an important role on the beneficiary perception of an IE 'doing its job' (Augusto and Muñoz 2019), so we need to acknowledge these somehow. As indicated above, there are two important dimensions to beneficiaries' assessment of how well an IE is serving her/him: this is the difference between *BCE* and *BCP*. That is, the gap between what the beneficiary *B* expects and what actually receives (or at least perceives to be receiving). We can consider a function $BCE(p_i, s_j, c_k, b, t)$ which measures how a beneficiary (b) prefers (p_i) a contextualized (c_k) service (s_j) at a certain time (t), and a function $BCP(p_i, s_j, c_k, b, t)$ which measures how that beneficiary perceives the actual delivery of that service. Let us call this the *level of Service Achievement Satisfaction of an IE system for a beneficiary b at time t*:

$$SAS(IE, b, t) = \sum_{i=1..p;j=1..s;k=1..c} BCE(p_i, s_j, c_k, b, t) - BCP(p_i, s_j, c_k, b, t)$$

For example let us take the contexts described in Table 1, and assume context *Front_door_use2* was used once, *Going_to_bathroom2* was used three times, *Getting_up_AM_routine2* was used once, *Skipping_lunch2* was used once, *Being_at_Kitchen2* was used twice. The expectation from beneficiary *PB* could have been that the system provided all the services described in the *Effect* section of Table 1, hence we can say the priority level for all these was top priority and let

us indicate that as 100%. Let us suppose the actual experience was not so good, some lights during *Going_to_bathroom2* context did not turn on and *PB* had an interface to provide feedback to the system and indicated the experience was 75%. Then the level of Service Achievement Satisfaction for *PB* that day can be measured as follows. $BCP = 1x1 + 0.75x3 + 1x1 + 1x1 + 1x2 = 7.25$ when the expectation was that the system worked perfectly well in delivering all services in all contexts: $BCE = 1x1 + 1x3 + 1x1 + 1x1 + 1x2 = 8$.Therefore, $SAS(SmartHome, PB, today) = BCP - BCE = -0.75$ indicating a negative experience. These calculations are mere examples, different functions can be explored at theoretical and practical level to find different options, this is only an example to complement the understanding of the concepts introduced above.

The consideration of *BCE* and *BCP* as tools to assess the system-human relation opens up a number of interesting variations. One initial expectation could be that an IE system will continuously try to bring the experience as close to what the beneficiary has indicated:

$$|BCE(p_i, s_j, c_k, b, t) - BCP(p_i, s_j, c_k, b, t)| = 0$$

However, more adventurous colleagues may like to consider what if the system is allowed to make creative suggestions which differ from the initial requirements in the believe it will maximize beneficiary preferences and an option could be to grant the system permission to make certain suggestions in certain contexts. In which case it needs to be measured whether the beneficiary agrees that the service delivered, although different than initially indicated to the system, is a pleasant surprise or a good alternative. The considerations above can be generalized to a group of beneficiaries, where theories about individual preferences can coexist with group preferences. A candidate building block to achieve this could be ontologies, although as we stated earlier on, is not a forced choice. This overarching function provides a way to optimize a system with regards to the preferences associated with the whole group of beneficiaries, rather than an individual preferences. For example optimizing group safety, or group economy. Each beneficiary can have a personal partial order of preferences preferences (Augusto and Muñoz 2019; Oguego et al. 2018) and the system could also have a separate Global Preferences partial order, *GP*, which can help to optimize group experience, especially when the system has more than one alternative course of action. Such hybrid multiusers optimization function such as:

$$SASm(IE, B, t) = \sum_{i=1..p;j=1..s;k=1..c;l=1..b} \left(BCE(p_i, s_j, c_k, b_l, t) - BCP(p_i, s_j, c_k, b_l, t)\right) * GP(p_i)$$

which behaves in the way of the previous one when *B* is a singleton, otherwise it moderates the impact of each individual preference by the value of the system dimension in the global scale. Of course all sort of strategies can be

created by refining the above in several directions, this is for future exploration. For example, a "cost to achieve a service" parameter can be added so that beneficiaries satisfaction is achieved favoring options with a lower cost. Also we left time undetermined as we discussed before this can be instantaneous or durative and is another interesting system dimension; however, one we cannot explore in detail in this article.

Conclusions

This article addressed some of the long-overdue challenges associated with the understanding of contexts within subareas of computer science concerned with the development of context-aware systems. Context is an implicit concept in daily life, it is so embedded in the real world that, like it happens with time, we use them without thinking much about them at a conscious level.

Given context acts as a technical "wild card" in language there is a lack of agreement on basic concepts, of commonly accepted and used methodologies and tools. This leaves increasingly popular technical areas which rely heavily on this concept trapped into a technical plateau of producing systems in an ad hoc way. There has been a plethora of systems produced however little has emerged in the way of reusable methods, tools or standards. A more systematic approach is needed for the area to progress in a more discipline oriented and scientific manner.

There have been extensive surveys of the use of context in relation to these technological area, all these previous work shed some light on this multi-dimensional and slippery concept. However none of them succeeded in producing something akin a theory that we can use to discuss the essential concepts which make up the fabric of contexts and its use by system developers. Here we attempted such enterprise at a level which is more formal than popular surveys, in a way that is not implementation dependent and in a way that fundamentally tries to highlight the key concepts which may be of relevance to the developers. This work is consciously aimed at the midway level in between the more generic surveys and the more ad hoc specific systems as this is where our area is lacking resources. We lack concepts and theories which have been discussed and allow the community to agree on which are the fundamental components and processes which should be given more energy and attention when engineering systems of this kind.

We started our journey by reassessing the concept at a more informal level where we highlight that the ultimate goal of systems in this area and the metric to their success rests on the intended beneficiaries' satisfaction. We then transfer this values to a more formal outline of the basic components and the operations which emerge as relevant. We highlighted the tasks of context activation, comparison, influence, construction, and interaction. We hinted at how they may work and explained these through examples, admittedly simple

given space restrictions. We not only emphasized the importance of the beneficiaries and the services to produce the effects that matter, we also highlighted the possibility and the importance of escalating that into a multi-users level. Our theory allows to express simply and clearly how preferences and expectations of the various users can be taken into consideration and the performance of the system can be linked to overall satisfaction and how effectively the system can exhibit optimized behavior.

Although context is a construct humans use frequently in daily life without the need of a deep understanding, this is not so straightforward when we want to use it to create systems on which humans well-being depend on. Hence a deeper and more careful understanding is required to create more effective context-aware systems. This study addresses this need and provides a first step of several necessary steps within our scientific community to converge to a set of good principles to follow. Equally, there is interesting work happening about context in other areas of science, some of which were mentioned at the beginning, and it will be constructive if these efforts interchange knowledge.

Disclosure Statement

No potential conflict of interest was reported by the author(s).

References

Aarts, E., and R. Roovers. 2003. IC design challenges for ambient intelligence. In *Proceedings of the Conference on Design, Automation and Test in Europe*, Munich, Germany, 10002–07. USA: IEEE Computer Society.

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26 (11):832–43. doi:10.1145/182.358434.

Atzoria, L., A. Iera, and G. Morabito. 2010. The internet of things: A survey. *Computer Networks* 54 (15):2787–805. doi:10.1016/j.comnet.2010.05.010.

Augusto, J. C., A. Aztiria, D. Kramer, and U. Alegre. 2017. A survey on the evolution of the notion of context-awareness. *Applied Artificial Intelligence* 31 (7–8):613–42. doi:10.1080/08839514.2018.1428490.

Augusto, J. C., and A. Muñoz. 2019. User preferences in intelligent environments. *Applied Artificial Intelligence* 33 (12):1069–91. doi:10.1080/08839514.2019.1661596.

Augusto, J. C., D. Kramer, U. Alegre, A. Covaci, and A. Santokhee. 2018. The user- centred intelligent environments development process as a guide to co-create smart technology for people with special needs. *Universal Access in the Information Society* 17 (1):115–30. doi:10.1007/s10209-016-0514-8.

Augusto, J. C., J. Giménez-Manuel, M. Quinde, C. Oguego, M. Ali, and C. James- Reynolds. 2020. A smart environments architecture (Search). *Applied Artificial Intelligence* 34 (2):155–86. doi:10.1080/08839514.2020.1712778.

Augusto, J. C., V. Callaghan, D. J. Cook, A. Kameas, and I. Satoh. 2013. Intelligent Environments: A manifesto. *Human-centric Computing and Information Sciences* 3 (article 12). doi:10.1186/2192-1962-3-12.

Augusto, J. C. 2001. The logical approach to temporal reasoning. *Artificial Intelligence Review* 16 (4):301–33. doi:10.1023/A:1012551818243.

Augusto, Juan C. (2003). A General Framework for Reasoning about Change. New Generation Computing 21 (3): 209–246. Springer.

Bazire, M., and P. Brézillon. 2005. Understanding context before using it. In *Proceedings of the 5th International and Interdisciplinary Conference Modeling and Using Context*, ed. A. K. Dey, B. N. Kokinov, D. B. Leake, and R. M. Turner, vol. 3554, 29–40. Paris, France: Springer. Lecture Notes in Computer Science.

Beck, H., M. Dao-Tran, and T. Eiter. 2018. LARS: A logic-based framework for analytic reasoning over streams. *Artificial Intelligence* 261:16–70. doi:10.1016/j.artint.2018.04.003.

Benerecetti, M., P. Bouquet, and C. Ghidini. 2000. Contextual reasoning distilled. *Journal of Experimental & Theoretical Artificial Intelligence* 12 (3):279–305. doi:10.1080/09528130050111446.

Benerecetti, M., P. Bouquet, and M. Bonifacio. 2001. Distributed context- aware systems. *Human–Computer Interaction* 16 (2–4):213–28. doi:10.1207/S15327051HCI16234_06.

Bettini, C., O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni. 2010. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing* 6 (2):161–80. doi:10.1016/j.pmcj.2009.06.002.

Bouquet, P., C. Ghidini, F. Giunchiglia, and E. Blanzieri. 2003a. Theories and uses of context in knowledge representation and reasoning. *Journal of Pragmatics* 35 (3):455–84. doi:10.1016/S0378-2166(02)00145-5.

Bouquet, P., F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. 2003b. C-OWL: Contextualizing ontologies. In *Proceedings of the Second International Semantic Web Conference*, ed D. Fensel, K. P. Sycara, and J. Mylopoulos, vol. 2870, 164–79. Sanibel Island, FL, USA: Springer. Lecture Notes in Computer Science.

Brewka, G., S. Ellmauthaler, R. Gonçalves, M. Knorr, J. Leite, and P. Jörg. 2018. Reactive multi-context systems: Heterogeneous reasoning in dynamic environments. *Artificial Intelligence* 256:68–104. doi:10.1016/j.artint.2017.11.007.

Brewka, G., T. Eiter, M. Fink, and A. Weinzierl. 2011. Managed multi-context systems. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, ed. T. Walsh, 786–91. IJCAI/AAAI, Barcelona, Spain.

Cabalar, P., S. Costantini, G. De Gasperis, and A. Formisano. 2019. Multi-context systems in dynamic environments. *Annals of Mathematics and Artificial Intelligence* 86 (1–3):87–120. doi:10.1007/s10472-019-09622-0.

Callaghan, V., G. Clarke, A. Pounds-Cornish, and S. Sharples. 2000. Buildings as intelligent autonomous systems: A model for integrating personal and building agents. In *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS'06)*, Tokyo, Japan.

Cardelli, L., and A. D. Gordon. 1998. Mobile ambients. In *Proceedings of 1st International Conference on Foundations of Software Science and Computation Structure, FoSSaCS'98, Lisbon, Portugal*, ed. M. Nivat, vol. 1378, 140–55. Springer. Lecture Notes in Computer Science.

Chen, H., T. Finin, and A. Joshi. 2003. An ontology for context-aware pervasive computing environments. *Knowledge Engineering Review (New York, NY, USA)* 18 (3):197–207. September.

Dao-Tran, M., and T. Eiter. 2017. Streaming multi-context systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, ed. C. Sierra, 1000–07. IJCAI, Melbourne, Australia.

Dertouzos, M. L. 2002. Human-centered systems. In *The invisible future*, ed. P. J. Denning, 181–91. New York, NY, USA: McGraw-Hill, Inc.

Dey, A. K. 2001. Understanding and using context. *Personal and Ubiquitous Computing* 5 (1):4–7. doi:10.1007/s007790170019.

Dey, A., and G. Abowd. 1999. Towards a better understanding of context and context-awareness. *Computing Systems* 40 (3):304–07.

Ellmauthaler, S., and Jörg Pührer. 2015. Asynchronous multi-context systems. In *Advances in knowledge representation, logic programming, and abstract argumentation - essays dedicated to Gerhard brewka on the occasion of his 60th birthday*, ed. T. Eiter, H. Strass, M. Truszczynski and S. Woltran, vol. 9060, 141–56. Springer. Lecture Notes in Computer Science.

Forbes, G., S. Massie, and S. Craw. 2020. Fall prediction using behavioural modelling from sensor data in smart homes. *Artificial Intelligence Review* 53 (2):1071–91. doi:10.1007/s10462-019-09687-7.

Galton, A., and J. C. Augusto. 2002. Two approaches to event definition. In *Proceedings of the 13th International Conference Database and Expert Systems Applications*, ed. A. Hameurlain, R. Cicchetti, and R. Traunmüller, vol. 2453, 547–56. Aix-en-Provence, France: Springer. Lecture Notes in Computer Science.

Giunchiglia, F. 1993. Contextual reasoning. *Epistemologia Special Issue on I Linguaggi E Le Macchine XVI*: 345–64.

Gu, T., X. H. Wang, H. K. Pung, and D. Q. Zhang. 2004. An ontology-based context model in intelligent environments. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Diego, CA, USA, 270–75.

Guha, R. V. 1991. *Contexts: A formalization and some applications*. Stanford, CA, USA: Stanford University.

Hagras, H., M. J. Victor Callaghan, G. C. Colley, A. Pounds-Cornish, and H. Duman. 2004. Creating an ambient-intelligence environment using embedded agents. *IEEE Intelligent Systems* 19 (6):12–20. doi:10.1109/MIS.2004.61.

Hamblin, C. 1972. Instants and intervals. In *The study of time*, ed. F. Fraser and G. Muller, 324–28. New York: Springer Verlag.

McCarthy, J. 1993. Notes on formalizing context. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, ed. R. Bajcsy, 555–62. Morgan Kaufmann.

McCarthy, J. 1990. Artificial intelligence, logic and formalizing common sense. In *Philosophical logic and artificial intelligence*, ed. R. Thomason., and K. Academic. Dordrecht, 161-190. https://link.springer.com/chapter/10.1007%2F978-94-009-2448-2_6

Nedopil, C., C. Schauber, and S. Glende. 2013. *AAL stakeholders and their requirements*. Ambient Assisted Living Association, Brussels, Belgium. https://www.aal-europe.eu/wp-content/uploads/2015/02/AALA_Knowledge-Base_YOUSE_online.pdf

Obrst, L., and D. Nichols. 2005. Context and ontologies: Contextual indexing of ontological expressions In *Proceedings of the AAAI 2005 Workshop on Context and Ontologies*, Menlo Park, CA, USA: AAAI Press.

Oguego, C. L., J. C. Augusto, A. M. Ortega, and M. Springett. 2018. Using argumentation to manage users preferences. *Future Generation Computer Systems* 81:235–43. doi:10.1016/j.future.2017.09.040.

Perera, C., A. B. Zaslavsky, P. Christen, and D. Georgakopoulos. 2014. Context aware computing for the Internet of Things: A survey. *IEEE Communications Surveys & Tutorials* 16:414–54. doi:10.1109/SURV.2013.042313.00197.

Pradeep, P., S. Krishnamoorthy, R. K. Pathinarupothi, and A. V. Vasilakos. 2021. Leveraging context-awareness for Internet of Things ecosystem: Representation, organization, and management of context. *Computer Communications* 177:33–50. doi:10.1016/j.comcom.2021.06.004.

Preuveneers, D., and Y. Berbers. 2005. Automated context-driven composition of pervasive services to alleviate non-functional concerns. *International Journal of Computing and Information Sciences* 3 (2):19–28.

Ranganathan, A., R. E. Mcgrath, R. H. Campbell, and M. D. Mickunas. 2003. Ontologies in a pervasive computing environment In *Proceedings of the Workshop on ontologies and distributed systems*, Acapulco, Mexico.

Santofimia, M. J., F. Moya, F. J. Villanueva, D. Villa, and J. C. López. 2008. An agent-based approach towards automatic service composition in ambient intelligence. *Artificial Intelligence Review* 29 (3–4):265–76. doi:10.1007/s10462-009-9145-2.

Siewe, F., H. Zedan, and A. Cau. 2011. The calculus of context-aware ambients. *Journal of Computer and System Sciences* 77 (4):597–620. doi:10.1016/j.jcss.2010.02.003.

Stavropoulos, T. G., D. Vrakas, D. Vlachava, and N. Bassiliades. 2012. BOnSAI: A smart building ontology for ambient intelligence. In *2nd International Conference on Web Intelligence, Mining and Semantics*, ed. D. D. Burdescu, R. Akerkar, and C. Badica, vol. 30, 1–30. Craiova, Romania: ACM. 12.

Stavropoulos, T. G., D. Vrakas, and I. P. Vlahavas. 2013. A survey of service composition in ambient intelligence environments. *Artificial Intelligence Review* 40 (3):247–70. doi:10.1007/s10462-011-9283-1.

Stefan, I., C. L. Aldea, and C.-S. Nechifor. 2018. Web platform architecture for ambient assisted living. *Journal of Ambient Intelligence and Smart Environments* 10 (1):35–47. doi:10.3233/AIS-170470.

Turner, R. M. 1998. Context-mediated behavior for intelligent agents. *International Journal of Human-Computer Studies* 48 (3):307–30. doi:10.1006/ijhc.1997.0173.

Turner, R. 1993. Context-sensitive reasoning for autonomous agents and cooperative distributed problem solving In *Proceedings of the IJCAI Workshop on Using Knowledge in its Context*, Chambery, France.

Weiser, M. 1991. The computer for the 21st century. *Scientific American* 265 (3):94–104. doi:10.1038/scientificamerican0991-94.