Context and Adaptivity in Pervasive Computing Environments: Links with Software Engineering and Ontological Engineering

Ahmet Soylu[1,2], Patrick De Causmaecker[1,2], Piet Desmet[1]
[1]K.U. Leuven, Interdisciplinary Research on Technology Education and Communication (iTec), Kortrijk, Belgium

[2]K.U. Leuven, Combinatorial Optimization and Decision Support (CODeS), Kortrijk, Belgium
Email: {Ahmet.Soylu, Patrick.DeCausmaecker, Piet.Desmet}@kuleuven-kortrijk.be

***Abstract*—In this article we present a review of selected literature of context-aware pervasive computing while integrating theory and practice from various disciplines in order to construct a theoretical grounding and a technical follow-up path for our future research. This paper is not meant to provide an extensive review of the literature, but rather to integrate and extend fundamental and promising theoretical and technical aspects found in the literature. Our purpose is to use the constructed theory and practice in order to enable anywhere and anytime adaptive e-learning environments. We particularly elaborate on context, adaptivity, context-aware systems, ontologies and software development issues. Furthermore, we represent our view point for context-aware pervasive application development particularly based on higher abstraction where ontologies and semantic web activities, also web itself, are of crucial.**

***Index Terms*— pervasive computing, context, context-awareness, adaptivity, semantic web, ontologies, software engineering.**

I. Introduction

Machines that fit the human environment instead of forcing humans to enter theirs will make using a computer as refreshing as taking a walk in the woods [1, 2].

Computing has already dispersed from dedicated and stationary computing units into the user environment and presently we are surrounded with mobile, multimodal and multiuser computing devices. [3] notes that pervasive computing (a.k.a. ubiquitous computing, ambient intelligence) takes advantage of distributed computing and mobile computing while inheriting problems (e.g. remote access, high availability, power management, mobile information access) in these fields increasingly. Apart from these problems, since they have been studied under related domains effectively, it is reasonable to say that we already achieved a lot as a part of Weiser's vision in the sense of hardware and network technologies by considering the advancements in the networking technologies, computing power, miniaturization, energy consumption, materials, sensors etc. [4]. However we are still far from the complete puzzle, pervasive computing is not just about developing such small computing residents for the real life, variety of applications exploiting such extended hardware infrastructure are the other side of the coin. Spreading computing all over life imposes new challenges which were already foreseen in this vision. Anywhere and anytime computing needs to cope with computing devices which are mobile, users which are mobile and software applications which are mobile. [5] partly referred to this mobility as "*constantly changing execution environment*"; we rather call it "*constantly changing computing setting*" which refers to mobility and dynamism of both related parties. Furthermore, heterogeneity of such environments hardens the challenges of such vision since software and hardware markets have already been populated with variety of applications and tools coming from different vendors.

Does this increasing digitization of life require more attention of people? This question, which originates from mobility and dynamism, requires achievement of the following approach:

The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are undistinguishable from it [1].

In other words seamless integration of computing into people's life is a must of the pervasive computing vision. If we don't want people to bother about the computing devices and the applications surrounding them, while they are making use of them, we need to make computing devices and applications to bother about people. Utilizing all these physical resources and synchronizing various applications available through this extended dynamic infrastructure for the benefit of the users requires an "*intelligence*" behind. This implies that computing systems need to reach a level of understanding of the settings in which they are being used, and the complex relations between the various elements of these settings. This ability is called "*Perception*", however this is one side of the coin. On the other hand computer systems

This article is based on "Context and Adaptivity in Context-Aware Pervasive Computing Environments", Soylu A., De Causmaecker P., Desmet P., 2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing, 7-9 July 2009, Brisbane, Australia, © 2009 IEEE.

need to be able to exploit this understanding by adapting their behaviors accordingly (i.e. that is to response properly according to perceived context), which is called "*Adaptivity*". These two interrelated challenges make pervasive computing diverge from mobile computing and distributed computing since the challenges needed to cope with are not strictly bound with these fields. Besides they are rather new and their theoretical grounding is not yet sufficiently mature. Moreover autonomous applications in such environments need to operate collectively in order to achieve maximum utility (at least to ensure a conflict free execution), however the heterogeneity of the pervasive environments hinders seamless integration of different applications and devices, that is, interoperability. Standard compliance is of prominent importance for such a requirement since standards help to ensure interoperability and five other important abilities: (1) *re-usability*, (2) *manageability*, (3) *accessibility*, (4) *durability* [6]. All in all, we define pervasive computing environments as follows:

...intelligent digital ecosystems which are seamlessly situated in user's physical environment. Such ecosystem is defined as a collection of seamlessly integrated, mobile/stationary and autonomous/non-autonomous devices and applications, where higher mobility and autonomy is of crucial. Intelligence for such systems is defined as capability of being able to perceive changing computing context and to response collectively in a proper manner (i.e. to adapt) for maximum user utility.

Accordingly, we consider perception, adaptivity, interoperability and standard compliance as key enablers of pervasive computing apart from other technical challenges, inherited from aforementioned fields, and social challenges (i.e. privacy, trust and security). Our main research is about enabling any-where and any-time adaptive learning environments which is highly dependent on pervasive computing vision. Despite the fact that this paper is based on a domain specific (i.e. e-learning) perspective in order to enable pervasive e-learning, the solutions and approaches we do aim to follow and propose are rather generic. In this paper, our contribution can be grouped under two categories: (1) *theoretical*, (2) *technical*. From theoretical point of view, we do extract and extend theoretical aspects found in the existing literature, and we work toward to integrate these ideas into a common understanding for the future pervasive computing systems. Regarding technical point of view, we review the related literature with the purpose of integrating and extending existing technical work which are generic, standard-based, and compliant with the overall understanding which we synthesized. Readers should bear in mind that our purpose is not to give an exhaustive review of the literature, but merely to provide a selective and integrative review of comparatively important and promising approaches found in the literature. Our selection criteria is particularly based on following parameters: (1) *standard-compliance* (although limitations of the available standards might hinder our efforts, since available standards are based on the characteristics of traditional computing, keeping available standards in the core of the development and research and to extend them when required is our guiding mantra), (2) *generalness*, (3) *applicability*, (4) *simplicity*, (5) *ease of development*, (6) *extensibility*, and (7) *scalability*.

The remainder of this paper is structured as follows: in section II, we introduce methodology and domain specific motivation of our research. We elaborate on the notion of context and its relation with adaptivity in section III, we further refer to characteristics and categorization of context in respective subsections. In section IV, categorization of context-aware systems is briefly referred while context management is elaborated in section V. We further investigate some key problems and basic solution approaches in section VI. In section VII, we introduce our view point for context-aware application development based on model driven and ontology driven approaches by referring related literature, we also emphasize use of World Wide Web as an information source for pervasive environments. Finally we conclude this paper in section VIII.

II. MOTIVATION AND METHODOLOGY

Challenges which are based on natural characteristics of pervasive computing systems (i.e. mobility, dynamism and heterogeneity) can be evaluated from a more domain specific perspective, that is, e-learning in our case. E-learning refers learning which uses variety of technologies such as internet, television etc. in a manner pointed out by [7]:

…e-enhancements of models of learning. That is to say that; using technology to achieve better learning outcomes, or a more effective assessment of these outcomes, or a more cost-efficient way of bringing learning environment to the learners [7].

E-learning evolved a lot by the emergence of computers and later internet, and continues its raise with the advancements in network and mobile services and software market which offers variety of advanced learning environments, tools and adaptive technologies. Apart from technological advancements, e-learning also faced with some important pedagogical movements particularly learner centric and self directed approaches which are based on constructivist learning theories. These approaches consider learners as active participants of the learning instead of passive consumers and change the role of teachers as facilitators who assist learners to clarify their goals and enable them to be capable of planning, executing and evaluating their learning progress and outcomes collaboratively, without taking a particular position in the discussions, rather than being pure source of information [8, 9]. [10] notes that providing active, stimulating, authentic learning experiences that support learner collaboration, construction and reflection is major challenge for success of e-learning. Such approaches triggered the creation of learner-centric, social and collaborative learning environments. Today embedding social networking and collaboration into learning progress is considered as driving force for learner's motivation and activity [11]. Moreover social software (e.g. blogs, wikis etc.) gained an important place for e-learning thus the mine of data, World Wide Web, because of Web 2.0's great collaborative potential, Wisdom of Crowds, and simple find-remix and share rule. As a consequence, e-learning market has already been over populated with such tools and platforms to support different types of learning communities with learning management, content management and communication tools [9]. Learners are not bound to neither individual learning environments, as closed box of pure information, nor to classical in-class learning environments anymore. Instead by the guidance of the constructivist theories they are facing with variety of tools including their particular learning environments which enables them to collaborate, to reach endless amount of information of web, and to remix-share it, thus also to create social networks. Depending on the case, these tools are being used individually by learners, or by means of mash-ups, or as heterogeneous systems which involve several tools and might be centered around a particular learning system [12]. Furthermore, with the emergence of pervasive computing vision learners and the learning process also goes for time, place and device independence, that is, learn anywhere-anytime.

Pervasive learning goes hand-in-hand with the idea of "*always on*" education and extends concepts of collaborative learning, cooperative learning, constructivism, information rich learning environments, self-organized learning, adaptive learning, multimodal learning, and a myriad of other learning theories [13]. Growing tool and device landscape and the pervasive computing vision, forces e-learning domain to adjust itself within this new landscape appropriately. Therefore there is also a line of research towards pervasive learning (a.k.a. ubiquitous learning) where a pervasive e-learning environment might be defined as a setting in which students can become totally immersed in the learning process [14]. Pervasive computing takes part in an experience of immersion as a mediator between the learner's mental (e.g. needs, preferences, prior knowledge), physical (e.g. objects, other learners close by) and virtual (e.g. content accessible with mobile devices, artifacts) contexts [15]. We work towards enabling different applications in such learning environments to be seamlessly integrated (i.e. to be interoperable) and to be aware of the setting which they are used and to collectively adapt their behaviors according to the available context information. Enabling computing settings where capabilities, requirement and characteristic of entities are known to each other decouples these entities, that is, independence which is required for mediation process, that is adaptation. Hence, we particularly list following basic interrelated requirements for such pervasive learning environments: (1) *device independence*: applications and data should be always accessible without any device dependence, (2) *application independence*: data should be always accessible without any application dependence, (3) *adaptivity and adaptability*: learning environment and elements of this environment should dynamically adapt according to context of learner(s) and users should be able to configure such environments such as composing/decomposing data and applications, (4) *collective operation*: applications in such environments must be able to collectively operate for the benefit of users in a seamless manner. Adaptivity is long studied both in adaptive web systems and adaptive e-learning systems [16], and in such systems adaptivity is generally considered as an aspect between user and application based on user profiles and models. However, although we do follow a user-centric approach, other requirements (1,2 and 4) make it necessary to broaden the adaptivity from learners to the whole environment in which user is engaged in order to be able to mediate between different independent entities of such settings. Although we do not claim to propose solutions for all the challenges of pervasive computing or pervasive learning, the approaches which we propose are common enough to be employed within generic pervasive environments. That is only possible by first providing a generic understanding (i.e. theory).

Briefly our research question can be formulated as follows:

How to enable adaptivity (in broader sense) in Pervasive Learning Environments through applying available context information?

Accordingly, our main approach is to integrate and extend available technical and theoretic approaches in pervasive computing, context-aware computing and adaptive systems literature into e-learning. In this stage we mainly focus on constructing a theory which represents overall framework of our understanding and to which our future practice should comply with. The theory that we focus on is broad while the practice is limited in the scope (i.e. e-learning) which is based on constructed theory. Therefore many of the challenges introduced in this paper are either in our long term agenda or merely mentioned for the attention of other researchers. Challenges specific to our main research is subject to another publication. The overall approach is depicted in Fig. 1, the lower domains are much more generic and theory intensive in order to constitute overall frame of our research. The upper domains are more specific and dependent on lower domains, innovative aspects of the research increases towards specific domains while integrative aspects are higher in more generic domains. In this stage of our work, we mostly focus on the theoretical and technical aspects of context-aware pervasive computing and adaptivity (in a generic sense) in such environments, that is, first two levels of our research pie.

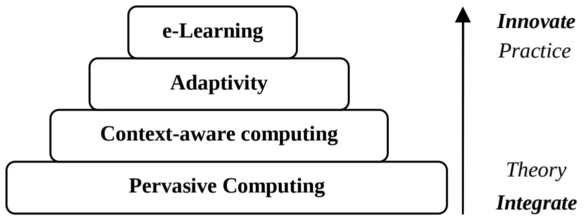

Figure 1. Fitting Boxes – uLearning Research Pie: abstract view of the research required for enabling anywhere and anytime adaptive learning environments - innovation through integration.

Adaptivity (in a more specific sense) and e-learning is subject to another in depth research where specifics of the domain and existing work (i.e. e-learning, adaptive e-learning) need to be elaborated based on the theory and practice introduced in this paper.

III. CONTEXT AND ADAPTIVITY

The notion of context is of crucial for pervasive computing systems, it is a central notion for context-aware pervasive computing environments as we already mentioned in section I. Indeed, according to the view represented previously, pervasiveness, context-awareness and adaptivity are bound to each other, that is, one implies the other one. The notion of context has, over time, been extensively discussed in the literature [17, 18, 19, 20, 21, 22]. [23] reviews related work and after briefly criticizing the concept, author gives the well known definition of context:

Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between the user and application, including the user and applications themselves [23].

Previous definitions of context in the literature usually refer to context as location, identity of users, and nearby people. Intuitively it is reasonable to accept location, identity, activity and time [18, 23] as important elements of context. However these elements are not sufficiently broad to cover the notion. The definition given in [23] is more generic and open-ended and covers context as a whole. The reason why it is not possible to give a more specific definition is the openness of the notion of context; a particular knowledge is considered to be context information in one setting while it is not part of context in another setting. [24] points out that it is not always possible to enumerate a priori a limited set of context that matches the real world context and [25] also refers to the same issue by pointing out that it is not possible to enumerate all important aspects of a situation. Therefore, by following the definition of [23], we are lead to conclude that defining the scope for context should leave an important role for context-aware application development rather than providing an exhaustive definition of context.

Since it is not possible to predefine all the dimensions of context, then how can we decide whether a piece of information can be counted as context or not? [26] remarks that context of use will have a substantial impact on the appropriate behavior of applications, without being a primary input source. Well, then imagine an automatic door which uses a sensor to detect presence of a person in front for switching between its states (i.e. close or open). Location of the user is indeed primary input for this particular system, and obviously the application for this system is primarily designed to sense the situation (i.e. presence of a person) and to act accordingly. Several example applications can be listed where context information is used to adapt application behavior without being primary input of the application in contrast to the previous example. Then, should we also consider primary inputs of applications as context information? Or should we only consider context as the information which is not primary input of the application but which characterize the situation? Here is another example: consider a word predictor application for speaking-impaired people [27]. This application can use previous user inputs to predict the word which the user presently tries to type. Here primary input of a previous context turns out to be another context dimension. Indeed every application, whether we consider it context aware or not, is designed for a specific and restricted context of use. Therefore these applications provide a particular set of behaviors for a fixed context of use. Hence, we are lead to conclude that context awareness is ultimately related with adaptivity. It is based on exploiting recruited context information and to adapt its behavior accordingly. In order to consider a piece of information to be context, it has to be ensured that this piece of information enables the corresponding application to modify its behaviors with respect to this piece of information and its relation with other context dimensions. [28] states:

[...] something is context because of the way it is used in interpretation, not due to its inherent properties. The voltage on the power lines is a context if there is some action by the user and/or computer whose interpretation is dependent on it, but otherwise is just part of the environment [28].

The mostly used context dimension is location, however it is a known fact that context is not limited to the location and physical objects in the environment. Consider the field of the Adaptive Web [16]. Much work has been done to define user models (e.g. user knowledge, goals etc.) and user profiles (e.g. user interests etc.) in order to enable web applications to act accordingly. Such adaptivity includes adaptive presentation, adaptive information filtering etc.. User profiles and user models are also type of context which are abstracted to a higher level mainly from logs of applications with which users interact. This implies that context information does not necessarily require to be gathered by sensors. Finger prints of users (i.e. application logs, web logs etc.) collected by applications can also be exploited to reach high level context information. Adaptive web applications belong to the field of Adaptive Systems and indeed they eventually are useful in the field of context-aware and pervasive computing systems. This relation implies that most of the practice and knowledge constructed in this field may be applied to the context-aware systems. Eventually it is the application which needs to adapt. Applications primarily need to adapt to the user, however the environment that the user lives in and the devices that are in contact with the user in turn influence the user. Applications adapt to the user as wells as to the environment, the devices and the complex relationships among each other. This is the result of mobility and dynamism of aforementioned peers. In an arbitrary setting where each device has its own characteristics and resources like screen size, CPU power, memory size, available input and output devices etc., applications need to adapt according to the context of the devices (i.e. resource awareness) to better serve the users [29, 30]. Adaptivity should not be understood as a one-to-one relation between user and application, in a pervasive computing setting, rather it should be considered as a relation between application and other elements of such settings (e.g. devices, physical environment, users etc.). Pervasive computing considered to be the third wave in the computing where first wave is main frame computing – one computer for many users-, second wave is personal computing – one computer per user -, and third wave is the one where many computers available for one user. Indeed the later wave (i.e. pervasive computing) should be considered more broadly, that is, many computers for many users. That makes computing much more sophisticated from application development point of view, since applications are not only required to accommodate needs of only one user but of many users, that is, to adapt masses.

As a conclusion, context is an open concept since it is not limited with one's imagination. Any system that exploits available context information needs to define the scope for the context. Adaptivity is the primary relation between computing and context, and to count any information as context we need such a relation. Any system can focus on any context category (in particular to the user(s)). However, we need to be aware of that the application needs to adapt to various context dimensions although it also has its own context dimensions.

A) Characteristics of the Context

In the previous section instead of focusing on the definition of context, we rather tried to comment on context from different perspectives to give a deeper insight into the notion. We now investigate some specific characteristics.

First of all, context is "*dynamic*" [24, 31, 32, 33]. Although some context dimensions are static like the name of a user, most of the context dimensions are highly dynamic like the location of a user. Furthermore, some context dimensions change more frequently than others. One dimension may change its state every second while another dimension only changes its state every year, this also implies that context is temporal [33, 34]. What is more important to see is the evolving nature of context, i.e. it is "*dynamically constructed*" [32]. Consider user knowledge: it evolves dynamically over time, i.e. user adds new knowledge pieces to his knowledge or some knowledge is forgotten. These changes in state do not require destruction of previous states, but the states evolve. Therefore [32] suggests not to support a particular context but to support the evaluation of context:

[...] not to use of predefined context within ubiquitous computing system, but rather how can ubiquitous computing support the process by which context is continually manifest, defined, negotiated, and shared [32].

It is intuitively evident that several context dimensions are somehow interrelated [32, 33], that is, context is "*relational*". For instance, there are different kinds of relations between people in your home and in your job. Your being at home or in office is normally related with present time. Perception is not just about realizing concepts but also about understanding relations between these concepts which are necessary to interpret situations and behaviors. Relationships between context dimensions thus hold an important place for both context representation and interpretation.

[35] points out that computational systems are good at gathering and aggregating data and humans are good at recognizing contexts and determining what is appropriate. The computer system level of understanding and recognition is limited, hence computer systems are far from recognizing situations properly. Besides, it is a known fact that even human beings sometimes are unable to understand/evaluate the exact situation. That is what we call misunderstanding. Hence even for a given well modeled closed domain (i.e. a closed set of real world data), a computer system might lack proper perception. This is related to imperfection of context information, that is, context is "*imperfect*": ambiguity, irrelevance, impreciseness and incompleteness of context dimensions [33, 34, 36]. Consider the context information acquired via sensors. It is a known fact that sensors do not provide hundred percent of accuracy. Besides, multiple sensors might provide different readings for the same context value. How can one really judge a student's knowledge based on his answers to a multiple choice exam? Can one logically decide that it is night by simply considering the light level?

B) Categorization of the Context

It is possible to categorize context in various ways by considering different characteristics of the context. These categorizations are useful both for application development and for understanding of the context. [33] notes that classifying context is useful for managing quality of context, for instance dynamic context elements are prone to noise. Moreover such classifications are also useful for context modeling, in early conceptual phases and later, and they are required to define some specifics of adaptivity and context management (e.g. abstraction).

Acquired raw context information usually requires a certain level of abstraction which will be discussed briefly in section V. However, for a short insight, consider the example of location: a sensor might sense location as coordinates whereas the application might require this information in a more abstract way like the name of the city. Therefore location information based on coordinates requires to be abstracted in order to be comfortable with the application. Hence it is possible to categorize context from the application point of view [37] into (1) *low level context information* (a.k.a. implementation context) and (2) *high level context information* (a.k.a. application context). Low level context information is usually sensed by sensors or collected by means of application logs. [33] considers low level context information as environmental atomic facts. High level context information is derived from low level context information. However these are implicit means of collecting low level context information. It is also possible to gather context information explicitly, e.g. asking the user to provide context information directly.

[33] suggests that the ideal case is placing fewer demands on user attention (i.e. less direct user interaction).

Context can also be categorized from the collection point of view [38] which is indeed related with the above categorization: (1) *direct* (sensed or defined), (2) *indirect* (by means of inferring from direct context). Direct context refers to the collection of context information without realizing any extra processing of the gathered information. If the information is gathered implicitly by means of sensors, it is called "*sensed context*". If the information is gathered explicitly, it is called "*defined context*". We already mentioned that sensors are not the only means of collecting context information, application logging is just another way to do so. Therefore we propose to further categorize sensed context as "*sensor based*" and "*application based*". Direct context refers to low level context and indirect context refers to high level context information according to the previous categorization.

Context information can be categorized from a temporal point of view into two categories: (1) *static context*, and (2) *dynamic context*. Static context does not change by time like gender or name of a person. Dynamic context keeps changing in different frequencies depending on the context dimension like your location or age. This implies that for a dynamic context dimension, various values might be available. Hence, management of temporal character of context information is of crucial either in the sense of historical context or in the sense of validity of contextual information available.

Apart from categorizing context based on characteristics of the context, [5] categorizes context based on grouping similar context dimensions into: (1) *computing context*, (2) *user context*, and (3) *physical context*. Later [39] extends this categorization with (4) *time context*. [40] provides a similar context categorization; (1) *physical context*, (2) *social context*, and (3) *internal context*. [41] provides another categorization which includes (1) *infrastructure context*, (2) *system context*, (3) *domain context*, and (4) *physical context*. These categorizations are usually at higher granularity, hence they do not reveal enough information about themselves, and this might limit their usefulness for development of context-aware systems. Moreover some of them are more application oriented, hence the categorizations are not well balanced. We propose eight categories for context aware settings where we want to achieve an optimal granularity and want to represent main actors (i.e. entities) of a typical pervasive computing setting in a more real-world oriented manner. This categorization provides a clear layering for context-aware system development and may serve as an initial step toward a generic conceptualization. We argue for a layered categorization of context without considering any taxonomical relation: (1) *user context* (internal, external), (2) *device context* (hard, soft), (3) *application context*, (4) *information context*, (5) *environmental context* (physical, digital – e.g. network -), (6) *time context*, (7) *historical context*, (8) *relational context*. It is important to know what application is in use in which device, in which environmental setting, and at what time by which user etc.. Therefore context varies as a product of dimensions under disclosure of these context categories. User context splits into "*external user context*" and "*internal user context*". External user context is easier to sense (e.g. name, gender, height, and weight etc.) while internal user context is harder to sense [40] (e.g. user feelings – hate, love etc. -). Internal context might be derived by interpreting diverse low level context information such as blood pressure, hormone levels etc. Considering the device context, we distinguish "*hard device context*" and "*soft device context*" where hard device context refers to the physical properties of the device (e.g. CPU, memory etc.) and soft device context refers to the available software components in the device etc.. Application context refers to capabilities and requirements of an application, e.g. target platform, memory requirements etc.. Concerning environmental context, we distinguish "*physical environment context*" and "*digital environment context*". Physical environment context covers the real world entities and their characteristics such as nearby objects and their identities while digital context refers to the digital entities such as network capabilities. Information resides in digital space together with applications, context adaptive access of information is of crucial part of computing, particularly for the web environment. Hence information context refers to properties of meaningful information pieces available in different formats (e.g. text, image etc.), it is surprising to see that information has not been considered as an independent entity either in available context categorizations or various context models in the literature (to the best of authors knowledge). Time usually refers to time of situation, time zone, part of the day etc.. Historical context refers to situations that occurred before based on the temporal characteristic of context. Relational context refers to relationships between the different context dimensions, that is, it aggregates and represents different types of relations between the elements of a particular context-aware setting. Although, relations have been used in context conceptualization, they have not be considered as a context entity explicitly, we advocate that it is worth to consider relations as an contextual information since they also characterize the situation of an entity. Historical context elements and relationships among context elements (this is relational context) are important for interpreting the situations. We previously mentioned that proposed categorization may serve as a generic conceptualization (i.e. upper ontology) for our future context model. In Fig. 2 and Fig. 3 a rough conceptualization is depicted with some possible immediate sub-entities.

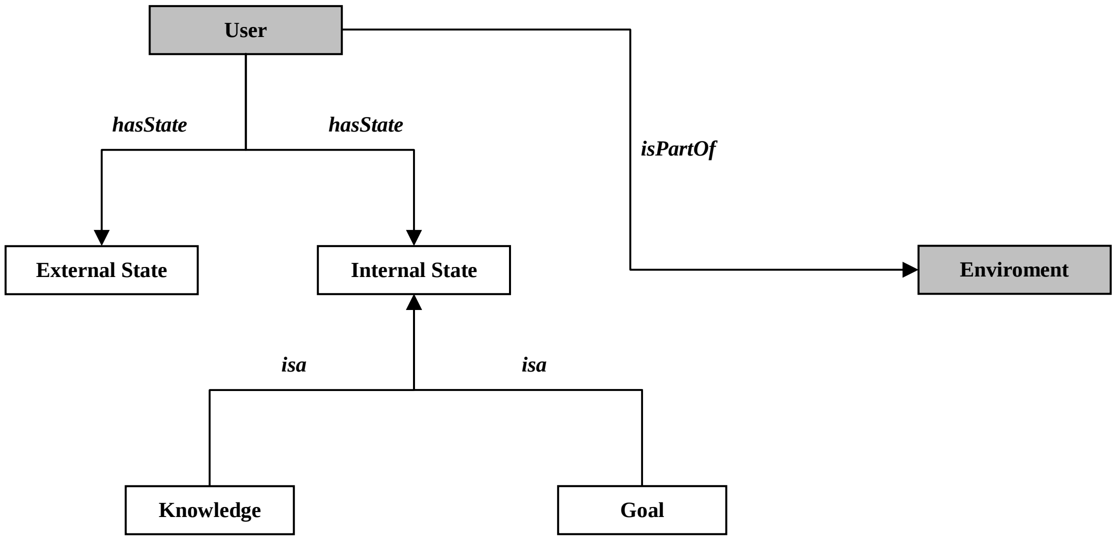

Figure 2. First part of the proposed upper context conceptualization, external state and internal state concepts has been shown as a part of user concept where user concept is part of environment.

[42] notes that generic uniform context models are more useful. Although there are already some proposals for a generic context models in the literature (see section V.), our rough proposal provides clear advancements similar to previously mentioned context categorizations such as optimal granularity and balanced representation of actors. Secondly information has been shown as an independent entity, and as a main actor of context which has been omitted in previous conceptualizations and categorizations, importance of such approach is detailed in section VII. This initial conceptualization only defines the borders of our understanding of context, a more elaborate formalized conceptualization (i.e. ontology) is to be developed where previous context models and standardized vocabularies are to be re-used.

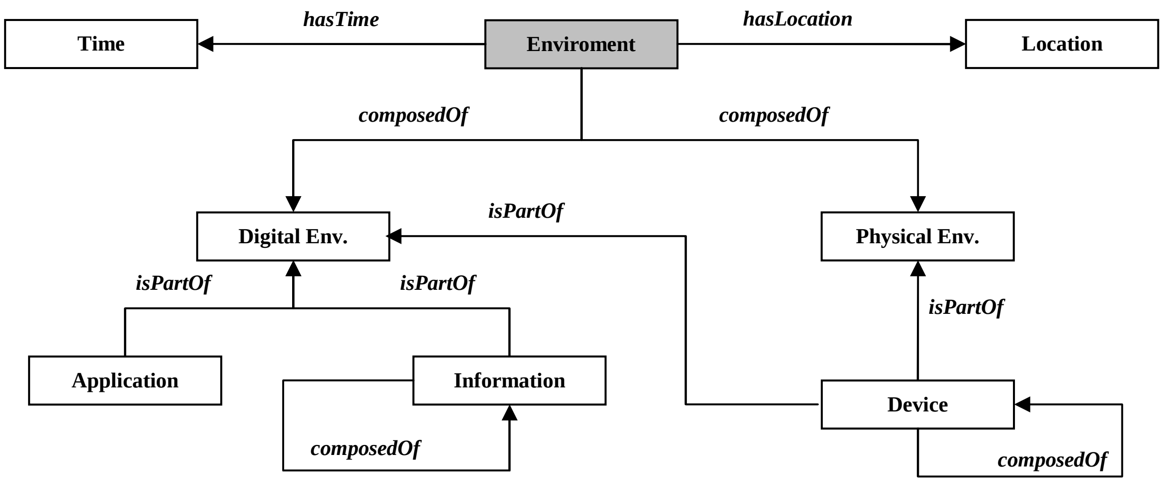

Figure 3. Second part of proposed upper context conceptualization, environment concept is composed of digital environment and physical environment concepts.

We can further group aforementioned context categories into technical and non-technical context for the sake of separation of concerns. Non-technical context includes context categories which are not related with the technical aspects such as internal user context, while the technical context involves context categories related with the technical aspects of context such as device context, and digital environment context. Although there is no straightforward way to distribute previous context categories into technical and non-technical context folds, non-technical context categories are mainly domain specific and require to be identified by domain experts, e.g. for pervasive learning environments, an expert is required to identify context categories or individual context dimensions related with learning aspects.

IV. CONTEXT AWARE SYSTEMS

In previous sections, context in pervasive computing has been reviewed. In what follows we elaborate on the definition of context-aware computing as it has been discussed in e.g. [17, 22, 23, 40, 43]. Earlier definitions usually involve a loose enumeration of context dimensions (e.g. location, nearby people etc.), and the later ones often concatenate on the relation between computing, context and user. It is clear that context needs to be employed to better serve the users, such point is already commonly noted in definitions like:

[… context aware computing] aims to enable device to provide better service for people through applying available context information [40].

Above a generic definition of context aware computing is given, which emphasizes the relation between user, context and computing, but how do we apply available context information? Although various categorizations for context-aware systems are already given [5, 23, 43], we prefer to re-interpret these categorizations based on adaptive systems, particularly according to adaptive web systems. This is because we defined adaptivity as a key factor of intelligence and as a key relation between context and computing for context-aware computing systems. Therefore by referring to [5, 23, 43] and the field of adaptive web [16] for categorization of context-aware computing applications, we propose below categorization: (1) *context based filtering and recommendation of information and services*: examples might include finding the nearest printer, accessing the history of a nearby object etc., (2) *context based presentation and access of information and services*: e.g. selecting voice when screen displays are not available (multimodal information presentation and user interfaces), dynamic user interfaces etc., (3) *context based information and service searching*: e.g. location aware query rewriting for a search for available restaurants (query rewriting is a technique used in adaptive web systems for information filtering by rewriting a user query according to the user profiles) etc., (4) *context adaptive navigation and task sequencing*: adaptive navigation is a technique employed in adaptive web systems. We can extend this idea in pervasive computing since a user's interaction might consist of several related sub-tasks in relation with his goals and might lead to context aware task sequencing, (5) *context based service and application modification/configuration* : this need mainly arises from different devices available in the environment, e.g. disabling particular features depending on the capabilities of target device, (6) *context based actions*: [44] proposes three levels of context dependent automatic actions: manual, semi-automatic, and automatic. [45] notes that fully automatic actions based on context are rarely useful, and incorrect actions can be frustrating, (7) *context based resource allocation*: this might include allocating physical recourses (e.g. memory, even non-hardware physical resources) for the use of other entities in the setting (e.g. applications, users etc.).

It is worth to note that, adaptive behaviors of context-aware systems are not necessarily need to depend on the current context, rather such systems should also be able to adapt proactively by making use of current context or historical context to predict future context of the setting. An example is given in [46] where a user walks through the building and submits a printing request, the selected printer should not depend on the user's current location but rather to his final destination. According to presented categorizations and elaborations, we extend previous definition of context-aware systems as follows:

Context aware computing aims to enable better service delivery through proactively adapting use, access, structure and behavior of information, services, applications and physical resources with respect to available context information.

Above categorization also stresses the applicability of several techniques and methods in the field of the adaptive web as we already mentioned previously. Other interesting examples might be applications of collaborative filtering, mass adaptivity, case based adaptivity etc. in context aware systems. Collaborative filtering is the process of filtering or evaluating items using the opinions of other people [16]. Since pervasive computing systems are able to interact with different people in different context settings, they can use captured information for collaborative filtering, case based recommendation, and these systems can employ adaptivity for masses which are sharing common characteristics (e.g. understandings, behaviors etc.) in common pervasive computing settings. [47] is an example which provides recommendations by comparing users with other users in pervasive computing systems. Furthermore, a case based reasoning example is provided by [48], proposed methodology is to abstract raw context to user situation, to generate current user's case, and to provide adaptive behaviors by semantically comparing user's current case with other previously stored cases and corresponding behaviors of the system.

As a final remark, pervasive computing environments do not necessarily fully automate their behaviors where such behaviors can be in varying granularity as shown in Fig. 4 [49].

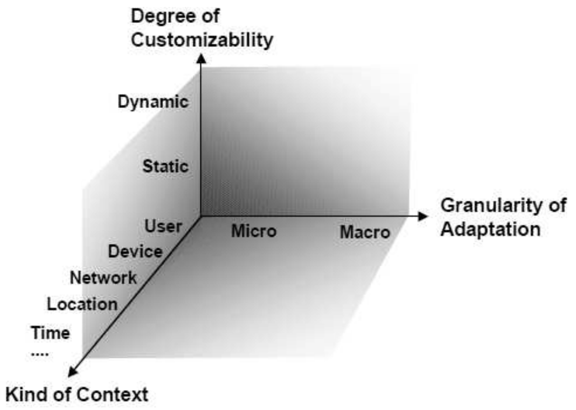

Figure 4. Space representation of context based Adaptation-Customization [49].

Such environments should also allow users to customize structure and behaviors of their environment (i.e. user control). Pervasive systems might facilitate such customization by means of adaptive guidance where environment does not automatically act or force user to one action but rather provides users with the required contextual information and recommendations. That is, adaptive behaviors do not necessarily need to result in "must"s or "have-to"s but in many cases also in "should"s and "might"s to give users a degree of control with the possible directions and their reasoning behind. In other words, in the scale of dynamic and static system adaptation, enabling users to control the environment does not imply that contextual information is useless for such a case. Rather, system can extend the limits of contextual information perceivable by the user's physical capabilities by serving contextual information gathered by sensors to the users rather than automatically adapting itself. An up-to-date and specific example is a famous social networking website, Facebook. This web application provides users with the contextual information of their network (by means of notifications) like who watches, reads what or who becomes friend with whom. In this way users can identify people with similar likes and arrange their own environment accordingly. Such case is also of use in the domain of e-learning, a system can provide users with the contextual information of the environment and other learners like who read what, who knows what, who takes the same courses or who works on the same problem, so learners can find appropriate mentors or construct a learning path for themselves. Such approach might be called as "*environment awareness*" for users which is counterpart of context-awareness for machines.

V. CONTEXT MANAGEMENT

We identify following groups of components for context management infrastructure by adopting [50] and [51] as shown in Fig. 5 which are required for realization of context-aware adaptation: (1) *context modeling and representation*, (2) *context capturing* (sensing), (3) *context abstraction and reasoning*, and (4) *context dissemination* (access and querying).

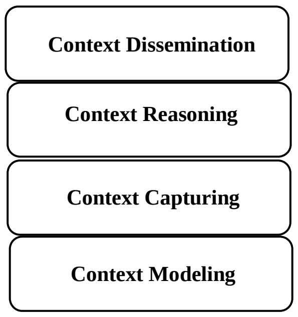

Figure 5. Components of context management infrastructure: context modeling, context capturing, context reasoning, and context dissemination.

Context capturing is handled by applications and physical sensors. [52] classifies sensors into following categories: (1) *physical sensors* which are hardware sensors available through physical environment to deliver physical measurements, (2) *virtual sensors* which are based on information and logs captured by the user applications and, (3) *logical sensors* which are based on reasoning various contextual information to produce higher level context information. Context dissemination is strictly related with the architecture of the context-aware application. Context information might be stored in central context brokers/blackboards, e.g. [42, 53, 54], or every application might hold its own contextual information, that is, context information might be distributed, e.g. [55, 56, 57] . Furthermore a hybrid approach might be possible where common contextual information is centralized and every application holds its specific contextual information. In all cases it is reasonable to call own-managed context information as "*local context*", context information managed by other entities as "*remote context*", and context information managed by central brokers as "*central context*" by extending the understanding presented in [58]. The most commonly used methods for context dissemination are *push* and *pull* mechanism [54, 59, 60, 61, 62]. In push mechanism applications register themselves to remote context entities or the central context brokers in order to be updated whenever a context of interest changes or is added. In pull mechanism, applications actively pool the remote and central context entities to check availability of context of interest, this might be possible by submitting synchronous or asynchronous query requests to the remote or central context entities. Within the same application, similar mechanisms can be employed, either by registered context listeners, e.g. [63], which triggers actions or asserts new contextual information, or by an ad-hoc manner where application itself checks the state of particular context information according to active execution stage.

Although distributed context management is researched by several users in the literature, e.g. [55, 56, 57], complexity and low efficiency of such approaches for the real-time systems do not seem to be promising yet. Resource-limited devices can hold their own contextual information, however even for limited amount contextual information reasoning can be time consuming for such devices (see section VI) or even not possible according to available resources. Considering e-learning, e-learning environments are complex, and variety of contextual information might be of use, hence presently we would prefer to use context broker architecture where reasoning, privacy and security, dissemination of contextual information are handled by such central architectures. A promising example is given in [53]. Such approach is of great use for the real-time, reasoning intensive applications. Scalability issues might arise in such architecture, for such a case using several powerful context-brokers can be of immediate solution.

In the following sub sections we do elaborate on the context modeling and representation, and context abstraction and reasoning respectively.

A) Context Modeling and Rrepresentation

Applications become perceptive when they maintain the model of its occupants and activities and user is only willing to accept an intelligent environment offerings services implicitly if he understands and foresees its decisions [64]. Furthermore, [65] notes that it is hard to re-use and change context information embedded into functional modules. Today's traditional intelligent computing is based on either ad-hoc AI techniques (e.g. data-mining, machine learning etc.), or based on hard-coded enumeration of possible contexts of use. However pervasive computing opens up infinite context space where it becomes hard to manage bindings between infinite context and behavior spaces (i.e. adaptive behaviors). Hence in order to enable computers to decide on (i.e. reason) adaptive actions (i.e. automatic, semi-automatic, manual) through automated reasoning and/or mediation processes - which requires to construct a bridge between humans and computers by enabling them to share a common world model - computing systems need to maintain a formal model of the settings in which they are being used and the complex relations between the various elements of these settings.

Several machine learning techniques (e.g. Bayesian networks, fuzzy logic etc. [50, 66]), statistical methods [67], and ontolgies as an AI paradigm can be used to model contextual information. [36] analyses several approaches in the literature according to data scheme used and concludes that ontologies are promising for context modeling. They represent explicit, formal (i.e. machine understandable) and shared conceptualization of real world aspects [68]. [69] refers to several reasons in order to use ontologies for context modeling: (1) *knowledge sharing*, (2) *logic inference*, (3) *knowledge re-use*. Considering context representation based on ontologies, [70] lists the following requirements for context representation: (1) *structured*, (2) *interchangeable*, (3) *composable / decomposable*, (4) *uniform*, (5) *extensible*, (6) *standardized*. There are several techniques to represent ontologies. We adopt categorization provided in [71] into: (1) *AI based*, (2) *software engineering* (e.g. UML), e.g. [33], (3) *database engineering* (e.g. ER, EER), and (4) *application oriented techniques* (e.g. key-value pairs), e.g. [31]. Software engineering techniques and database engineering techniques are limited in expressivity, i.e. they are not capable of expressing heavyweight ontologies (i.e. ontologies which model a domain with more constraints and expressiveness) but rather capable of modeling lightweight ontologies (i.e. ontologies which model a domain in a less expressive way and with less constraints). Indeed software engineering and database engineering are highly related with abstracting and modeling real world phenomena and logics into computer applications for a restricted context of use. This restriction causes software engineering and database engineering techniques to fall short when modeling generic context information. However it is not surprising to see that several software methodologies are well suited for ontology development (e.g. ontology re-engineering and software re-engineering [71]). AI based techniques are capable of representing high level ontologies, techniques based on frames and first order logic are mainly used. OWL (Web Ontology Language) [72], which provides a syntax and knowledge representation ontology, appeared as a prominent ontology formalization (i.e. representation) language with the advent of the semantic web. OWL is capable of representing main components of an ontology like classes (i.e. concepts), relations, instances and attributes. Since OWL is among the AI based techniques, it is suitable for high level ontologies. It can express complex relations between concepts, it is capable of acquiring dynamic information. Furthermore strong reasoning techniques and tools based on OWL provide a mean to deal with ambiguity in context. Hence it is reasonable to state that it is capable of capturing characteristics of context and criteria listed by [70]. There are already various works in the literature which employs ontologies, examples include [40, 43, 44], in order to maintain a context model and to apply reasoning over this model.

There are various tools and standards in the domain which supports ontology development and use based on OWL. We refer to prominent ones in what follows. Protégé provides a graphical interface to develop OWL based ontologies, JENA provides a semantic web framework where different ontology querying languages such as SPARQL and RDQL, and reasoning support are available. Semantic web rule languages such as RuleML and SWRL are already available and supported by various tools, which are used to describe logic rules.

B) Context Abstraction and Reasoning

We previously mentioned that context information is categorized as low level (i.e. implementation level) and high level (i.e. application level) context information. Low level context information is usually sensed by sensors or might be acquired from application logs. Afterwards it requires to be abstracted to the high level context information. According to [37] this happens in three ways: (1) *one-to-one*: one low level context value matches one high level context dimension, (2) *context fusion*: several low level context values match one high level context dimension, (3) *context fission*: one low level context value matches several high level context dimensions. Accordingly, we prefer to define context abstraction as process which asserts new contextual information by processing available context information.

We refer to [55] for an analytical understanding of context abstraction. We incorporate low level context – high level context mapping approaches given in [37] and [55] (see Fig. 6). [55] defines "*application space*" (i.e. or in broader sense: context space, C) as the universe of discourse in terms of available contextual information for an application and defines subspaces which reflect the real life situations within application spaces, which are, "*situation spaces*" (S). Authors further define "*context state*" as collection of context attributes' (i.e. dimensions) values at time t. Each context dimension have a "*value space*" (V_n) where value spaces represent range of values that a particular context dimension might have (e.g. 01 to 100 for age of a user). These value spaces might have discreet number of qualitative or quantitative elements or might represent a continuous range (discretization required). According to [55], some context dimensions might have greater importance than other context dimensions for a specific situation, therefore a weight is needed to be defined for each context dimension in each particular situation. Furthermore, authors note that for a particular situation, every context dimension can only match to some accepted values in its value space, and each accepted value in this set might have a different level of importance for this particular situation. Therefore every accepted value for a particular context dimension in a particular situation should have a different weight assigned (e.g. number of people in a room: 40 people should add greater contribution than 10 people would add for the situation of having a party, for example, where number of people in a room might vary from 10 to 50 for the situation of having a party). Moreover, some situations in situation space consists of combination of other situations (i.e. sub-situations). In order to have a consistent terminology we advocate the following understanding by re-interpreting [55]. Context information which maps to an adaptive behavior is a situation where a situation might be abstracted from low and high level context information and from other situations. A single atomic context dimension is low level context information where high level context is abstracted from low level context information and from other high level context information. High level context information does not map to any adaptive behavior but to the situations. Adaptive behavior represents both actions (manual, automatic etc.) and the change in application's normal flow and structure (e.g. adaptive presentation, recommendation etc.) based on the context. Accordingly we prefer to define context reasoning as a function which maps situations to adaptive behaviors. It is worth to note that abstraction can also considered to be a reasoning process, however for the sake of simplicity and consistency we prefer such distinction.

According to Fig. 6, situation S_1 is abstracted by one to one match of context dimension c_1, and S_2 is abstracted by fusion of c_2 and c_3. c_2 affects several situations (i.e. S_2 and S_3), that is, fission. Furthermore, some situations in situation space consists of combination of other situations (i.e. sub-situations). For instance S_2, S_4 and S_n are sub-situations of S_5, since context dimensions of sub-situations are totally covered by S_5. However this also requires that situation S_5 and its sub-situations need to have same accepted values of their context dimensions. Weighting approach allows us to calculate confidence values for inferred situations. That means provides a way to deal with ambiguity of context information.

Projecting low level context information to the high level context information and mapping situation space to behavior space, that is building the relation between the context and adaptive behavior, are not usually straightforward. It is hard to handle these mappings for systems having huge application and behavior spaces. The difficulty also arises from the main characteristics of context as discussed before: context is a dynamic construct, it is relational and imperfect. Context abstraction and reasoning based on ontologies is mainly handled by rule sets (i.e. pre-defined or user defined [48]) and ontological reasoning, i.e. *subsumption* and *realization* [71, 73]. A typical system usually includes a knowledge base and a context reasoner. Knowledge base stores terminological knowledge (in a *T-box*) e.g. concepts, properties etc., and assertional knowledge (in an *A-box*), e.g. individuals. Subsumption determines subconcept-superconcept relationships of concepts occurring in a T-box where realization computes which a given individual necessarily belongs to [73]. Reasoner holds context transformation rules in order to abstract low level context information, and context-behavior binding rules which binds context dimension(s) into a particular application behavior [37, 73].

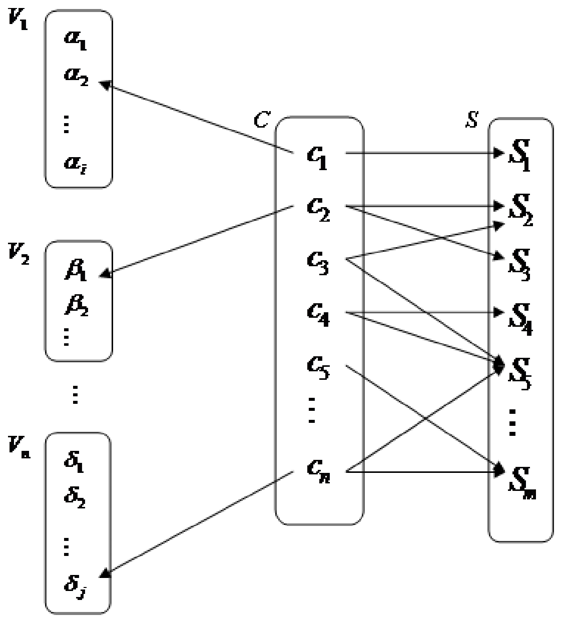

Figure 6. Context abstraction based on one-to-one, fusion, and fission approach. V sets represents value spaces for context dimensions, C represents context space while S represents situation space.

Considering adaptive behavior; rules which maps application space to behavior space (i.e. to the automatic behaviors) are usually pre-defined or user defined, as previously mentioned, that is called first order adaptation [74]. On the other hand if such rules are learnt by the system (i.e. through machine learning techniques), that is called as second order adaptation [74].

VI. KEY PROBLEMS AND BASIC APPROACHES

In this section, we will briefly refer to some key problems and basic solution approaches. Scope of this section is mainly limited with ontology based approaches, hence problems and solutions also are.

A) Dealing with Imperfectness

Imperfection of context has been studied by several researchers in the literature. Since basing automatic actions on imperfect context information is problematic, researchers usually refer to user involvement to decide on correctness of the context or actions (i.e. mediation), to detect inconsistencies or evaluate correctness of the context based on artificial intelligence techniques, e.g. [27, 35, 44, 45, 75, 76].

Considering other approaches in the literature, it is quite common to employ a metadata approach [77] to annotate acquired and derived context information with quality parameters. RDF reification is a common way of annotating ontologies based on OWL and RDF with quality parameters. However [78] notes that such approaches are not expressive enough to capture rich types of context information and to support reasoning. [79] list several metadata parameters for the quality of context: (1) *precision*, (2) *confidence*, (3) *trust level*, (4) *certainty*, (5) *granularity* and (6) *Uptodateness*, while [38] uses following similar quality measurements: (1) *accuracy*, (2) *resolution*, (3) *certainty* and (4) *freshness*. Uptodateness or freshness usually associated with an aging function based on a life cycle management approach where this aging parameter also affects the value of other quality elements like decreasing confidence or accuracy level depending on the freshness of the context information [57, 80, 81]. [78] notes that related approaches in the literature for reasoning about uncertainty with various metadata terms such as confidence and accuracy are not expressive enough to capture rich types of context information and support reasoning mechanism. Therefore authors decide to go for an integrated solution by combining Bayesian networks and ontologies, since ontologies are good at representing structural contextual information where Bayesian networks are good at representing probabilistic contextual information. Such approach combines probabilistic models for uncertainty and ontologies for knowledge reuse and sharing. Authors achieved their aim by adding new language elements to OWL and by creating a mapping through OWL model to Bayesian model.

Approach presented in [55], which has been introduced in section V, can be considered amongst more generic solutions. The introduced weighting approach allows us to calculate confidence values for inferred situations since every context dimension and every acceptable value for a contextual dimension is associated with a weight value. That means it provides a way to deal with ambiguity of context information. Authors also enable agents to merge or partition different perspectives of context which are managed by different agents in order to provide increasing level of accuracy. Another approach represented in [73] introduces means to handle irrelevant context dimensions where OWL ontologies are used as a representation formalism. It uses a context filter where authors define situation-action mapping as a policy. The more a policy is used, the more important it is. Authors use a weight recorder to record usage of policies where they eliminate irrelevant contextual information according to usage records of policies.

Since it is not possible to clean all the ambiguity of the context information based on artificial intelligence techniques, metadata approaches or many others, it is reasonable to use artificial intelligence techniques and others to some extent, and to employ a user mediation mechanism [27] for crucial situations, examples include [27, 75]. [76] emphasizes user involvement because user knows more, without user involvement system cannot evolve and system can lead wrong operations. The matter is enabling right level of balance between automatic actions and user mediation which should of course optimized based on priorities and importance of the situations.

B) Reasoning Performance and Managebility

Ontologies might grow up into huge knowledge bases which is problematic along with the heavy reasoning load for resource constrained devices in a pervasive environment [82, 83]. Through experiments [69] concludes that reasoning is time expensive, but still good for non-real-time applications. Authors identify three main performance factors: CPU speed, complexity of logic rules and size of context information.

In [69], authors suggest separating context use and reasoning where reasoning is done by resource rich devices and complexity of rule set need to be controlled. In a knowledge base there is a T-box which holds general concepts, their properties etc., and an A-box which holds individual specific information (i.e. instances). Tbox is usually static and classification and loading is time consuming in T-box [84, 85], hence it is usually loaded and classified offline [85]. There are numerous attempts in the literature to cope with this challenge like partial ontology fetching and evaluation, ontology encoding, synchronization and replication of ontologies etc. [30, 83, 86]. The most basic approach is to create plug-in (i.e. modular) ontologies, it is also beneficial from management point of view. [87] notes that modularity is the key requirement for large ontologies in order to achieve re-use, manageability and evolution. Usually there is an upper ontology (generic ontology) and a domain ontology (lower ontology, or plug-in ontology) [53, 62, 69, 88, 89], this approach enables corresponding domain ontologies to be plugged into a generic ontology based on the application domain. We further advocate that a domain ontology alone also might include considerable amount of irrelevant contextual information hence it needs to be further partitioned, it is reasonable to call these sub-partitions as task ontologies where the root element of such ontologies are called as active or master context element. An example might be as follows, a smart home domain ontology can be partitioned as bed-room ontology, kitchen ontology etc., where master context elements are "being in the bed-room", and "being in the kitchen" respectively. It is reasonable to say that identifying such active context spaces might be used to control size of T-box and logic reasoning. A possible approach might be using basic data-mining techniques over the condition set of the inference rules in order to partition context space. A similar approach is presented in [90], since only one context is active at any point in time the number of rules that have to be evaluated are limited. [61] remarks that A-box increase causes exponential increase in reasoning process time, hence only related items need to be collected at the time of reasoning [84]. [61] notes that subscribe (PUSH) method allows us to know what we need in A-box beforehand for Pre-selection. Approach presented in previous sub-section which focuses on eliminating irrelevant context information based on policy recorders [73] also enhances reasoning process according to the view presented in this section.

Performance of reasoning engines is also of crucial. [91] lists two types of inference engines which are database (DB) based and main memory (MEM) based. In main memory systems reasoning is done when the query is requested. They are more efficient but they lack scalability because of memory needs. DBMS based engines are slower but good choice when large and complicated knowledge is required, and they are scalable. [91] evaluates performance of following inference engines Minevra (DB), Hawk (DB), Pellet (MEM), Jena (MEM) based on a set of criteria, e.g. load time, query response time, query soundness and completeness etc.. Experiments lead authors to conclude that all the mentioned inference engines are far from being commercialized although Jena presents a better performance overall. Although research on enhancing performance of reasoners is challenging, only encountered example is [85] which employs prime numbers to encode concepts in an ontology for enhancing ontological reasoning (e.g. subsumption).

We refer to scalability and manageability issues briefly, prominently based on a database approach. [92] notes that database style management is much more scalable then ontologies however it is not standardized. Reasoning engines usually hold individuals and concepts in a specific format (e.g. RDF triples) which is usually not subject to be accessed directly by other users or applications, even this is possible, it is hard to manage. However database style of management allows other users and applications to access and manipulate data in a easy way (e.g. various views, query engines etc.). Contextual information is not only required for reasoning purposes, applications and users might also need to manipulate such information. For example, imagine set of questions (i.e. items) and answers which are given by students to these questions. They are stored in a DB, item difficulties can be considered as contextual information. In order to abstract item difficulties from set of answers given, a computational process is required which is difficult to apply through ontological representation of the data. [93] uses a hybrid approach based on using knowledge bases and databases however authors limit use of databases for static contextual information. We advocate that scalability and manageability of databases and reasoning support of knowledge bases need to be employed together. Therefore we propose following rough model which is inspired from SQI [94] which might be of use. Overall approach is depicted in Fig. 7. According to proposed model, contextual information should be kept in databases, and only required contextual information need to be loaded to knowledge base for reasoning purposes. A query interface enables various applications and agents to submit queries in different query formats (e.g. SQL, SPARQL, RDQL etc.) which is subject to arrangement between the application and the query interface. Query is mapped to local query language of the database or knowledge base and query results are returned back in a common format (RDF, XML etc.) which is also subject to arrangement. Application also can send a command to load related contextual information from database to knowledge base. In order to enable such approach, a wrapper need to maintain a mapping between knowledge base and database. Automation of such mapping is possible, we refer readers to section VII for details of such automation and mapping.

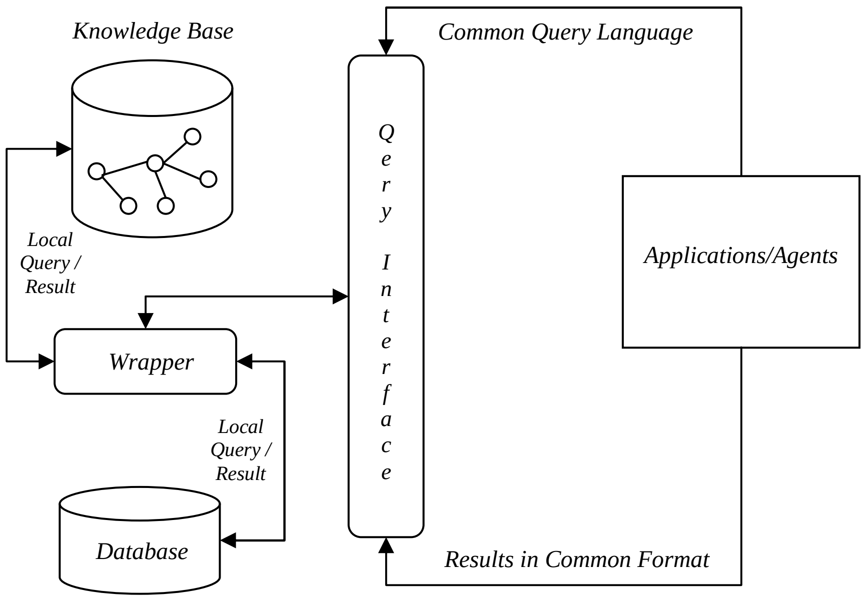

Figure 7. Merging relational databases and knowledge bases in order to enable scalable and efficient context reasoning and context management, only relevant contextual information is loaded for reasoning.

VII. TOWARDS A GENERIC APPROACH

[95] reports that context-aware applications are not yet come into market because of high development overheads, social barriers such as privacy and security, and an imperfect understanding of truly compelling uses of context-awareness. Furthermore several researchers remarks that context-aware computing lacks of appropriate infrastructure and middleware support, e.g. [38, 96, 97]. Hence, several research initiatives focused on developing such frameworks or middleware infrastructures based on various available software architectures, methods, techniques etc. , e.g. [53, 54, 57, 65, 98, 99].

Almost none of these developments or frameworks has been really considered as a killer application, they are usually based on context models and encapsulated common functionalities in one way or another way, e.g. agent based [98, 99], service oriented [57, 65], central brokers [53, 54] etc.. However approaches presented in the current literature of pervasive computing did not really manage to go beyond the borders of traditional computing and software engineering, although use of context models, particularly based on ontologies, can be considered as an important movement. Available studies employ ontologies for modeling and reasoning over context information, however we advocate that use of ontologies should be employed in every phase of software development, that is, both for separating reasoning logic and designing and specifying software artifacts. In other words we consider shift towards approaches based on higher abstraction as a key challenge in order to cope with increasing complexity of pervasive computing environments. Secondly, available approaches greatly undermine the place of World Wide Web (WWW) for tomorrow's pervasive computing environments. WWW is the biggest available digital layer of today and it is reasonable to claim that it will continue to be so tomorrow. Therefore, it is a fact that utilizing such a huge information source for pervasive computing environments are of great challenge. Fortunately, semantic web approaches which are already being used for context modeling and reasoning will be of great help. In subsection A and subsection B, proposed approaches are elaborated respectively.

All in all, researchers should re-construct and adjust software engineering approaches and use of WWW for pervasive computing environments which is tomorrow's computing indeed. In the following sub-sections we introduce our approach for these two mentioned challenges. Bear in mind that although the approaches we are going to mention are not purely novel since they are studied in their corresponding domains, our contribution is mainly based on synthesis and integration of such promising approaches based on a pervasive computing perspective.

A) Application Development

Apart from introducing new challenges pervasive computing also greatly catalyzes the problems inherent to the software development. Such challenges can be considered from development time and run time point of view. It is a known fact that maintaining knowledge of the application is essential since software development is subject to various changes. [100] refers to four fundamental forms of change: *personnel* (e.g. programmers, designers etc.), *development platforms*, *deployment platforms*, and *requirements*. Hence, in traditional application development, practically, that expert knowledge is lost; more accurately, that knowledge is embedded in code ready for architectural archaeology by someone who probably wouldn't have done in that way [101]! Therefore, a properly managed application knowledge ensures sustainability of the application by absorbing such changes. On the other side, pervasive computing requires computing entities in such environments to be aware of each other's characteristics and functionalities and to be able to communicate and share information in order to ensure collectivity. That is, assumptions done at the development time should be minimal and applications should be able to adjust themselves according to the various run time settings which might differ from each other in the sense of underlying technology, capabilities, requirements etc.. Approach presented in [102] reflects our understanding for context-aware pervasive application development (e.g. application context, hard device context, soft device context etc.). [102] simply considers devices as portals, applications as tasks, physical surroundings as computing environments. Based on this vision, authors divide the application life-cycle into three parts: design-time, load time and run time. Authors define criteria and models for each part. Considering design time, it is suggested that applications and application front-ends should not be written with a specific device in mind. Besides applications should not have assumptions about available services, therefore abstract user interfaces and abstract services need to be described. The structure of the program needs to be described in terms of tasks and sub-tasks instead of simply decomposing user interaction. Considering load time it is suggested that applications must be defined in terms of requirements and the devices must be described in terms of capabilities. Considering run-time, it is noted that it must monitor the resources, adapt applications to those resources and respond to changes. Such approach is based on higher abstractions of entities, including applications themselves. Indeed, that is how programming evolved from machine code assemblers to data structures, to object oriented languages and to the compilers in order to cope with increasing complexity. High-level languages replaced assembly language, libraries and frameworks are replacing isolated code segments in reuse, and design patterns are replacing project-specific code [103]. The next cycle of the abstraction, compelled by pervasive computing era, needs to reduce semantic gap between problem domain and representation domain based on higher abstractions of the business logic, application itself, and the reasoning logic based on contextual information. Conceptualizing the problem domain which is based on encapsulated abstract representations of entities, their capabilities, requirements, available functionalities and the complex relations between these entities and their characteristics will greatly reduce semantic gap with the representation domain and will isolate developers from the low level technical aspects of development. Ontologies might be considered as solution for such a higher level of complexity. [104] notes that it will be important to integrate ontologies with the software generation and management, perhaps using ontologies to semi-automatically generate interfaces. We further advocate using a top level of abstraction to automatically derive required software artifacts ranging from application code to specification, that is, letting programmers specify what programs should do rather how it should do it [99]. Moreover, ontologies can be used to automatically verify applications [105] before creating the code by means of using ontological reasoning process.

Early examples of context-aware pervasive applications are rather ad hoc, and are not based on a high level context models, hence reasoning support is not available, limited or hard coded. Examples include [31, 106, 107], although these systems did progress in various aspects of pervasive computing they are weak in supporting knowledge sharing and reasoning because lack of a common ontology [53]. Pervasive computing vision has opened up infinite context space which is required to be bind over infinite behavior space. Hence, it is hard to manage model of a setting and increasing number of rules in an ad hoc way, besides it is hard to share or reuse constructed knowledge which is directly hard coded into the application. Accordingly, latter applications are based on high level context models, particularly based on ontologies represented by OWL, RDF, UML etc., examples include [33, 62, 69, 70, 99, 108]. However existing work in the literature is mainly based on using traditional software engineering and computing paradigms in one or another way (i.e. various software architectures, encapsulating context management functionalities in various ways etc.), but far away from being revolutionary, and the novelty of contribution is almost limited with separating reasoning logic from application code. However, according to our perspective, ontologies need to be employed in every part of context-aware pervasive application development in order to enable higher abstraction. [109] notes that ontologies can be of use for (1) *communication between computer-computer, between human-human, between human and computer*; (2) *computational inference*; (3) *knowledge re-use and organization*. Communication between computer and computer addresses the interoperability problems, where human-human communication addresses the terminological ambiguity between developers and leads to a consistent framework for unification [110, 111]. One of the benefits of using ontologies is that they aid interaction between users and the environment since they concisely describe the properties of the environment and the various concepts used in the environment [104]. Particularly, enabling higher level of user-computer communication is of help for user mediation which is only possible when both entities share the same conceptual understanding of the setting. Furthermore, [112] points out that ontologies can be used for software engineering either at run-time or at development time. Having a knowledge base which is external to application for reasoning purposes is example of use of ontologies at run time (i.e. computational inference). Considering development time, a system can be specified and designed by the use of ontologies in a computing independent way, then designed ontology can be used to automatically generate application code, code skeletons (i.e. skeletal code) and other software artifacts such as database schema, UML diagrams etc.. Moreover, constructed knowledge is preserved and is ready to be re-used or to be shared. Development time use of ontologies is highly undermined by previous approaches for context-aware pervasive computing. Indeed a typical context ontology, by the nature of context, involves considerable amount of application knowledge. Therefore constructed knowledge should be used for automated code generation rather then re-modeling, re-defining and manually generating application.

Hence, we refer to related literature briefly for ontology driven development which can be employed for context-aware pervasive application development. [113] proposes a development method called “ontology oriented programming”, where the problem domain is expressed in the form of an ontology and such ontology is used to generate object oriented application code. This is programming paradigm is of a higher abstraction level than object-oriented programming (concepts versus objects), but which finally, through the indicated compiler, makes it possible to generate object-oriented code [113]. Although [114] sufficiently addresses the related literature for ontology-driven development, particularly at development time, Model Driven Development (MDD) [100, 101, 105] approach which is based on the same idea of automatically generating application code from models seems to be more mature. This is because of the experience, tools and standards available for this approach are more standardized and advanced. Prominently, Model Driven Architecture (MDA) [115], which is initiated by OMG consortium, holds an important place for MDD. MDA initiative offers a conceptual framework for defining a set of standards in support of MDD [104]. MDA software development life cycle includes a five step process [103]: (1) capture requirements in a *Computing Independent Model* (CIM), (2) create a *Platform Independent Model* (PIM), (3) transform the PIM to one or more *Platform Specific Models* (PSM) by adding platform specific rules and code that the transformation did not provide, (4) transform PSM to code, (5) deploy the system in a specific environment. UML standard which uses UML meta-model [101] is in the core of MDA for modeling. We previously mentioned that UML is considered to be a software engineering paradigm which can be used for representing ontologies, however it is only limited to represent lightweight ontologies. Therefore, UML approach in MDA might not be a proper choice to model both reasoning logic, application logic and contextual entities. Hence use of OWL and OWL knowledge representation ontology (i.e. ontology to represent an ontology, corresponds to UML meta-model) instead of UML and UML meta-model might satisfy our purposes. People use UML or object oriented languages because they are more close to development layer and might facilitate it, so OWL should also come closer to development layer [115]. This is possible by developing easy-to use visual development environments and tools. Accordingly, MDA process can be adopted to such ontology based approach as shown in Fig. 8 [117].

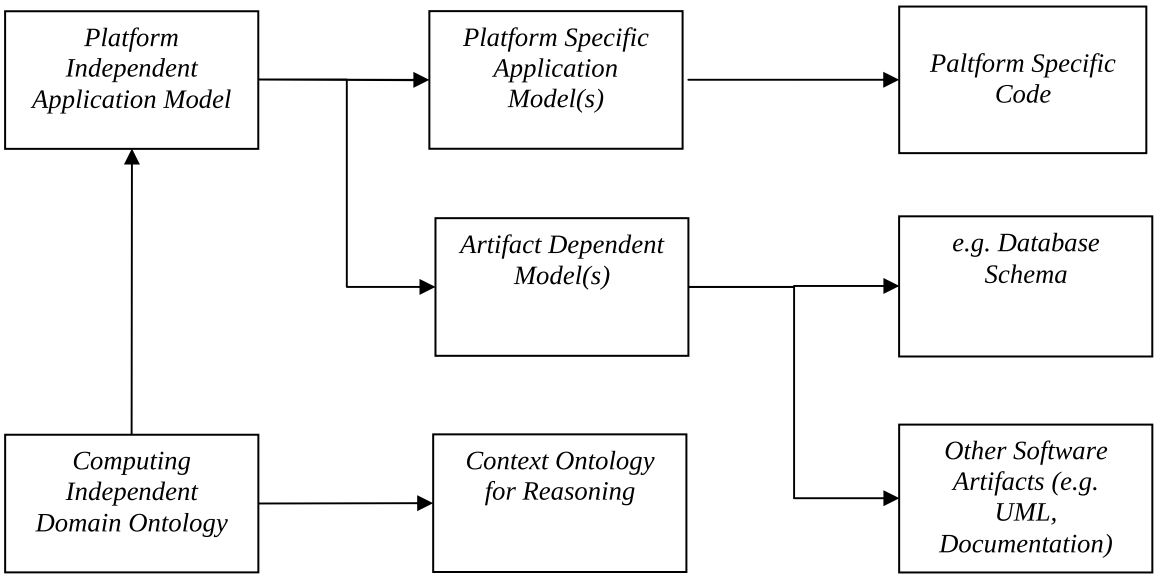
Platform Independent Application Model
Platform Specific Application Model(s)
Paltform Specific Code
Artifact Dependent Model(s)
e.g. Database Schema
Computing Independent Domain Ontology
Context Ontology for Reasoning
Other Software Artifacts (e.g. UML, Documentation)

Figure 8. An integrated abstract software development approach based on Model Driven and Ontology Driven Development where models are used both for automatic software artifact generation (i.e. development time) and for creating external reasoning logic (i.e. run time) [117].

In this approach, first a domain ontology need to be created, probably by applying one of the techniques [71, 118] for ontology development (omitted in the figure for the sake of brevity). Later, part of such ontology need to be employed for reasoning purposes, this is because not every element of this ontology need to be part of reasoning logic but rather part of application logic. Therefore, a *Platform Independent Application Model* (PIAM) is to be subtracted from the Domain Ontology. *Platform Specific Application Models* (PSAM) - e.g. JAVA, .NET etc. - and *Artifact Dependent Models* (ADM) need to be derived from PIAM. Finally, platform specific code, and various software artifacts need to be created by using PSAM(s) and ADM(s) respectively. Furthermore it might be required to fine-tune the code itself or to complete skeletal application code. Inserting handwritten code in MDA is especially important in MDA, because the process is both model-driven and iterative. That means that MDA tools are continually generating code [103]. Use of ontologies as a top level abstraction and to map it to different purpose-specific representations is supposed to enable rapid, sustainable application development which is quite suitable to the nature of application development for context-aware pervasive settings. An interesting example is presented in [78] where authors derive a Bayesian model from an ontology, that is, to merge Bayesian models and ontologies, for better reasoning in the sense of context quality. Such example clarifies what we do really refer by saying “purpose specific representation”, it does not necessarily need to be application code, or database schema, higher expressivity of ontologies enables them to be mapped less expressive representations. Complexity of pervasive spaces requires availability of different viewpoints of a model (e.g. a UML diagram might be more proper for documentation purposes, since it provides higher visual expressivity).

In this stage, it is required to have a brief look at available work in the literature which focus on ontology based automatic software artifact generation, particularly application code and database schema. [119] shows how to convert RDF schema and RuleML sources into Java classes, and [120] presents how to create a set of Java interfaces and classes from OWL ontology such that an instance of Java class represents an instance of a single class of the ontology with the most of its properties, class relationships and restriction-definitions maintained [120]. Similarly in [121], authors show how an OWL/RDF knowledge base can be integrated with conventional domain-centric data models (Enterprise Java Beans) and object-relational mapping tools (e.g. Hibernate). Considering relational databases, [122] presents a mapping technique from ontologies to relational databases in order to facilitate rapid operations (e.g. search, retrieval etc.) and to utilize benefits of relational management systems (e.g. transaction management, security etc.)

[97] points out that middleware must enable programmers to develop applications dynamically without having to interact with the physical world of sensors, actuators, and devices. In other words, we need a middleware that can decouple programming and

application development from physical space construction and integration. We further advocate that ontologies also have potential to enable users to program their own environment by means of synchronizing (e.g. sequencing, conditioning etc.) accessible services given by various entities (e.g. Outlook, TV, refrigerator etc.) of the environment. This is more than just being in the loop by means of meditation. We refer to such approach as "*environment programming*" which is only possible by enabling users and computers to share same conceptual understanding and enabling different entities to be plugged in the environment in a plug and play manner in order to advertise their available services.

B) Semantic Web

Pervasive computing enlarged traditional computing setting into the human layer of the earth. World Wide Web is the biggest digital information layer, hence it is unavoidable to stretch and integrate such an information layer over this new enlarged computing setting [123]. Two different approaches can be listed to merge web and pervasive computing environments, the first one is from application point of view where researchers use web as an application and communication space, that is, mapping is from real world to web as presented in [124] where real life objects have web presences. Second approach is from information point of view [123] where mapping is from web to real world which is our focus in this paper. Various researches in the literature use internet as an information source, and for many others use of internet might greatly advance their work (e.g. schedule information for smart spaces etc.), examples include [61, 99, 125]. Particularly, challenge can be identified as follows [123]:

…to enable variety of devices in pervasive computing environments to be able to extract and use valuable web resources by semantically structuring commonly published information chunks (e.g. events, user profiles etc.) and annotating various web documents with contextual information (e.g. size, format, requirements etc.) in order to enable adaptive retrieval and presentation of such documents.

Semantic web standards (e.g. OWL) have been used for varying challenges of pervasive computing systems such as context modeling as previously mentioned. However apart from its constructive existence in pervasive computing environments, semantic web activities, particularly embedded semantics [126], also have a crucial role for enabling pervasive computing environments to exploit web information. Different devices in pervasive environments connected to each other in various means such as wired and wireless networks, infrared connections, Bluetooth etc. Apart from these local ties, pervasive computing environments are also mostly connected with World Wide Web environment as shown in Figure 10 (a). Web environment is a huge information source, hence enabling pervasive computing environments to exploit valuable information in the web environment without imposing any extra burden is a challenging task which is of prominent importance. The semantic Web components such as XML, RDF, OWL etc. allows machine understandability of information however in explicit means. They are useful for system level aspects such as context modeling, or service messaging etc. and they aim at machine readability. However, RDFa [126], eRDF [128] and Microformats [129], embedded semantics, enables implicit annotation of information in web pages by using class attributes of (X)HTML elements which both provides human and machine readability of information. Accordingly, four layers of abstraction for information (including contextual information) can be identified (see Fig. 9) which is adopted from three layer of abstraction proposed by [34]: (1) *storage layer*, (2) *exchange layer*, (3) *conceptual layer*, and (4) *representation layer*. Representation layer, particularly embedded semantics, constitutes the missing link in current approaches.

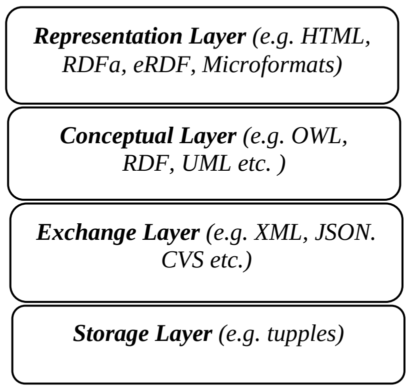
Representation Layer *(e.g. HTML, RDFa, eRDF, Microformats)*
Conceptual Layer *(e.g. OWL, RDF, UML etc.)*
Exchange Layer *(e.g. XML, JSON. CVS etc.)*
Storage Layer *(e.g. tupples)*

Figure 9. Four layers of abstraction for contextual information and information itself.: Storage Layer, Exchange Layer, Conceptual Layer, and Representation Layer.

With respect to the partial context model depicted in Fig. 10 (b), such approach particularly focuses on the information which is part of the digital environment together with the applications. This approach is also in line with the idea we mentioned in section III; context for information (i.e. information as an independent entity). Annotating web documents by contextual information or structuring commonly published information chunks by using proper and standard meta-data elements and vocabularies will enable context aware applications to filter, search and recommend and present these information pieces to users depending on the match between user context and information characteristics (i.e. context). An example is learning objects [12]; each might have different competence levels, media format etc. that might interest devices with different capabilities or users with different competence levels.

Structuring meaningful information chunks residing in the web environment and annotating various documents with their respective contextual information enables us to identify and retrieve information of interest easily by leaving out unnecessary content. Since information is decoupled from presentation, such approach also enables us to present retrieved information by making use of abstract user interfaces and multimodality.

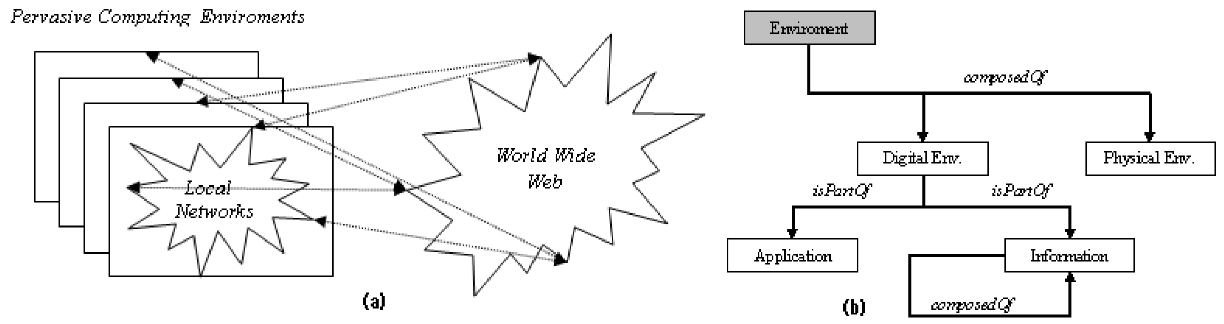

Figure 10. (a) Top level view of pervasive computing environments and World Wide Web, (b) partial view of our generic conceptualization.

VIII. CONCLUSIONS

Available research on context-aware pervasive computing lacks of a general understanding and a concrete methodology although required knowledge and vision are already distributed over respective literature. Particularly software engineering requires to be revised in order to cope with complexity which is introduced by pervasive computing. Proper understanding of context and its relation with adaptivity is of crucial in order to construct a new understanding for context-aware software development for pervasive computing environments. Role of the user in such environments need to be clearly understood in order to decide on right level of user control and automatic system behavior as well as for involving users for the mediation purposes. Moreover, pervasive computing expands the physical infrastructure of the digital environment which requires WWW to be properly coupled with such physical infrastructure as an ultimate information source.

Accordingly we first introduced our general understanding and methodology both in generic and domain specific sense (i.e. e-learning). Through this paper, we spend an effort towards integrating, and extending available theory and basic practice into a common understanding and into a conceptual framework which will lead us during both our long term and short term research. We elaborated on context and adaptivity by creating links with the application development issues. We referred a combined use of ontology driven and model driven approaches both at run time and development time where current practice is only limited with run-time use of ontologies. Such combined approach which is based on increasing level of abstraction might greatly facilitate rapid and sustainable application development. We further pointed out the semantic web approaches and web itself within the perspective of context-aware pervasive computing and indentified comparatively important challenges.

Our future work will be more focused on specifics of anywhere and anytime adaptive e-learning environments in compliance with general understanding and framework introduced in this paper. Consecutive complementary studies are expected to complete theoretical and technical framework constructed in this paper, that is, first two levels of our research pie depicted in Fig. 1. Further work will be based on specifics of adaptivity and context for e-learning domain which is indeed uLearning (i.e. ubiquitous learning) for our case. Our basic steps, more practically, will involve development of a generic and domain specific ontology for e-learning, and setting up required infrastructures and software components in order to make use of ontological reasoning and web information in such pervasive learning environments.

ACKNOWLEDGMENT

This paper is based on research funded by the Industrial Research Fund (IOF) and conducted within the IOF Knowledge platform “Harnessing collective intelligence in order to make e-learning environments adaptive” (IOF KP/07/006).

REFERENCES

[1] M. Weiser, “The computer for the 21st century”, Scientific American, pp. 94-98, 1991.

[2] M. Weiser, “Some computer science issues in ubiquitous computing”, Communications of the ACM 36, pp. 7:75-85, 1993.

[3] M. Satyanarayanan, “Pervasive computing: vision and challenges”, Personal Communications IEEE, Vol. 8, Issue 4, pp. 10-17, 2003.

[4] M. Bick, T. F. Kummer, “Ambient Intelligence and Ubiquitous Computing”, *In Handbook on Information Technologies for Education and Training*, Ed. Bernurs, P., Blazewicz J., Schmidt, G., Shaw, M., Springer Verlag, 2008.

[5] B. Schilit, N. Adams, R. Want, “Context Aware Computing Applications”, *Proceedings of Mobile Computing Systems and Applications*, pp. 85-90, 1994.

[6] MASIE Center e-Learning Consortium, “Making Sense of E-learning Standards and Specifications: A decision Makers Guide to their Adoption”, 2n Edition, The MAISE Center, November 2003.

[7] T. Mayes, S. de Freitas, “Review of E-Learning Theories, Frameworks and Models,” *JISC E-Learning Models Study Report*, Joint Information Systems Committee, www.jisc.ac.uk/elp_outcomes.html, 2004.

[8] S. Kuru, M. Nawojczyk, K. Niglas, E. Butkeviciene, A. Soylu, "Facilitating Cross-border Self-directed Collaborative Learning: The iCamp Case", *EDEN 2007 Annual Conference*, Naples, Italy, June 2007.

[9] B. Kieslinger, S. Fiedler, F. Wild, S. Sobernig, “iCamp: The Educational Web for Higher Education in an Enlarged Europe”, *eChallenges e-2006*, Barcelona, Spain, pp. 25-27, 2006.

[10] V. Wade, H. Ashman, “Evolving the Infrastructure for Technology-Enhanced Distance Learning”, Internet *Computing*, IEEE, Volume 11, Issue 3, pp. 16-18, 2007.

[11] T. Klobucar, “iCamp Space - an environment for self-directed learning, collaboration and social networking.”, *WSEAS Transactions on information science and applications*, vol. 5, no. 10, pp. 1470-1479, 2008.

[12] A. Soylu, S. Kuru, F. Wild, F. Mödrichter, “e-Learning and Microformats: a Learning Object Harvesting Model and a Sample Application”, *Proceedings of Mupple’08 Workshop*, Maastricht, Netherlands, pp. 57-65, 2008.

[13] S. Thomas, “Pervasive, persuasive eLearning: modeling the pervasive learning space”, *1st IEEE International Workshop on Pervasive eLearning (PerEL’05)*, pp. 332-336, 2005.

[14] V. Jones,and J. H. Jo, “Ubiquitous learning environment: An adaptive teaching system using ubiquitous technology.”, In R. Atkinson, C. McBeath, D. Jonas-Dwyer and R. Phillips (eds.), *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference*, pp. 468-474, 2004.

[15] A. Syvanen, R. Beale, M. Sharples, M. Ahonen, P. Lonsdale, “Supporting Pervasive Learning Environments: Adaptability and Context Awareness in Mobile Learning.”, *International Workshop on Wireless and Mobile Technologies in Education*, pp. 251-253, 2005.

[16] P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), *The Adaptive Web*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, 2007.

[17] B. Schilit, M. Theimer, “Disseminating active map information to mobile hosts”, *IEEE Network*, 8(5), pp. 22-32, 1994.

[18] N. Ryan, J. Pascoe, D. Morse, “Enhanced reality fieldwork: the context-aware archaeological assistant”, V. Gaffney, M. V. Leusen, S. Exxon (eds.), *Computer applications in Archaeology*, 1997.

[19] A. K. Dey, “Context-aware computing: the cyberdesk project”, *AAAI 1998 Spring Symposium on Intelligent Environments*, Tech. Report SS-98-02, pp. 51-54, 1998.

[20] D. Franklin, J. Flaschbart, “All gadget and no representation makes Jack a dull environment”, *AAAI 1998 Spring Symposium on Intelligent Environments*, Technical Report SS-98-02, 1998, pp. 155-160.

[21] T. Rodden, K. Cheverst, K. Davies, A. Dix, “Exploiting context in HCI design for mobile systems”, *Proceedings of Workshop on Human Computer Interaction with Mobile Devices*, 1998.

[22] R. Hull, P. Neaves, J. Bedford-Roberts, “Towards situated computing”, *Proceedings of 1st International Symposium on Wearable Computers*, pp. 146-153, 1997.

[23] A. K. Dey, G. D. Abowd, “Towards a better understanding of context and context-awareness”, *Technical Report GIT-GVU-99-22*, Georgia Institute of Technology, College of Computing, 1999.

[24] S. Greenberg, “Context as a dynamic construct”, *Human-Computer Interaction*, pp. 16:257-268, 2001.

[25] A. K. Dey, “Understanding and using context”, *Personal and Ubiquitous Computing*, Special issue on Situated Interaction and Ubiquitous Computing 5, 1, 2001.

[26] D. Chalmers, N. Dulay, M. Sloman, “Towards reasoning about context in the presence of uncertainity”, *Proceedings of Workshop on Advanced Context Modeling Reasoning and Management* (UbiComp'04), 2004.

[27] A. K. Dey, J. Mankoff, D. Gregory, “Distributed mediation of ambiguous context in aware environments”, Abowd and Scott Carter Proceedings, *Proceedings of UIST 2002*, 2002.

[28] T. Winograd, “Architectures for Context,” *Human-Computer Interaction*, Vol. 16, No. 2, 3 & 4, pp. 401-419, 2001.

[29] D. Preuveneers, Y. Berbers, “Towards context-aware and resource-driven self-adaptation for mobile handheld applications”, *Proceedings of ACM Symposium on Applied Computing*, Seoul, Korea, 2007.

[30] D. Preuveneers, Y. Berbers, “Encoding Semantic Awareness in Resource Constrained Devices”, *IEEE Intelligent Systems*, 23:2, 2008, pp 26-33.

[31] D. Salber, A. K. Dey, G. D. Abowd, “The Context Toolkit: aiding the development of context-enabled applications”, *Proceedings of Human Factors in Computing Systems (CHI '99)*, Pittsburgh, PA, pp. 434-441, 1999.

[32] P. Dourish, “What we talk about when we talk about context”, *Personal and Ubiquitous Computing*, 8(1),pp. 19-30, 2004.

[33] K. Henricksen, J. Indulska, A. Rakotonirainy, “Modeling Context Information in Pervasive Computing Systems”, *Proceedings of First International Conference on Pervasive Computing* (Pervasive 2002), pp. 79-117, 2002.

[34] R. Reichle, M. Wagner, M. U. Khan, K. Geihs, L. Lorenzo, M. Valla, C. Fra, N. Paspallis, G. A. Papadopoulos, “A comprehensive context modeling framework for pervasive computing systems”, *Proceedings of 8th IFIP International Conference on Distributed Applications and Interoperable Systems*, Oslo, Norway, 2008.

[35] T. Erickson, “Some problems with the notion of contextaware computing”, *Communications of the ACM*, 45(2), pp. 102-104, 2002.

[36] T. Strang, C. Linnhoff-popien, “A context modeling survey”, *Proceedings of Advanced Context Modelling, Reasoning and Management (Ubicomp2004)*, Nottingham, UK, 2004.

[37] W. Du, L. Wang, “Context-aware application programming for mobile devices”, *Proceedings of C3S2E'2008*, pp.215-227, 2008.

[38] X. Ying, X. Fu-yuan, “Research on context modeling based on ontology”, *Proceedings of CIMCA-IAWTIC’06*, 2006.

[39] G. Chen, D. Kotz, “A Survey of Context-Aware Mobile Computing Research”, *Technical Report*, TR2000-381, Dartmouth, 2000.

[40] L. Han, S. Jyri, J. Ma, K. Yu, “Research on Context-aware Mobile Computing”, *Proceedings of Advanced Information Networking Applications Workshops*, pp 24-30, 2008.

[41] A. Dix, T. Rodden, N. Davies, J. Trevor, A. Friday, K. Palfreyman, “Exploiting space and location as a design framework for interactive mobile systems”, *ACM Transactions on Human Computer Interaction*, 2000.

[42] M. Strimpakou, I. Roussaki, C. Pils, M. Angermann, P. Robertson, M. Anagnostu, “Context Modelling and Management in Ambient-aware Pervasive Environments”, *Proceedings of Int. Workshop on Location and Context-Awareness (LoCA2005)*, Oberpfaffenhofen, Germany, May 2005.

[43] J. Pascoe, “Adding generic contextual capabilities to wearable computers”, *Proceedings of 2nd International Symposium on Wearable Computers*, pp. 92-99, 1998.

[44] J. Mäntyjärvi, U. Tuomela, I. Kansala, J. Hakkila, “Context-studio–tool for personalizing context-aware applications in mobile terminals”, *Proceedings of Australasian Computer Human Interaction Conference*, Addison-Wesley Longman, pp. 64-73, 2003.

[45] P. Korpipää, J. Hakkila, J. Kela, S. Ronkainen, I. Kansala, “Utilizing context ontology in mobile device application personalization”, *Proceedings of Mobile and Ubiquitous Multimedia*, ACM Press, pp. 133-140, 2004.

[46] J. Coutaz, J. Crowley, S. Dobson, D. Garlan, "Context is Key", *Communications of the ACM*, 48(3), March 2005.

[47] C. Räck, S. Arbanowski, S. Steglich, "Context-aware, ontology-based recommendations.", *Proceedings of SAINTW'06*, 2006.

[48] Y. Bai, J. Yang, Y. Qiu, "OntoCBR: Ontology-based CBR in Context-aware Applications", *Proceedings of MUE 2008*, Korea, 2008.

[49] G. Kappel, B. Pröll, W. Retschitzegger, W. Schwinger T. Hofer, "Modeling Ubiquitous Web Applications – A Comparison of Approaches", *Proceedings of the Int. Conf. on Information Integration and Web-based Applications and Services (iiWAS)*, Austria, Sept. 2001.

[50] M. Khedr, A. Karmouch, 'Negotiating context information in context aware systems', *IEEE Intelligent Systems Magazine*, 19(6), pp. 21–29, 2004.

[51] T. Chaari, D. Ejigu, F. Laforest, M. Scuturici, "Modeling and Using Context in Adapting Applications to Pervasive Environments", *Proceedings of the IEEE International Conference on Pervasive Services (ICPS'06)*, pp. 111-120, Lyon, France, June 2006.

[52] J. Indulska, P. Sutton, "Location management in pervasive systems", *Proceedings of the Australasian Information Security Workshop (CRPITS '03)*, pp. 143–151, 2003.

[53] H. Chen, T. Finin, A. Joshi, "An ontology for context-aware pervasive computing environments", *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, 18(3), pp.197-207, 2004.

[54] P. Korpipää, J. Mäntyjärvi, J. Kela, H. Keränen, E. J. Malm, "Managing context information in mobile devices", *IEEE Pervasive Computing*, pp.42-51, 2003.

[55] A. Padovitz, W. S. Loke, A. Zaslavsky, "Multiple agent perspective in reasoning about situations for context-aware pervasive computing systems", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 38, No. 4, 2008.

[56] F. Perich, A. Joshi, T. Finin, and Y. Yesha, "On Data Management in Pervasive Computing Environments", *IEEE Transactions on Knowledge and Data Engineering*, October 2003.

[57] J. Bardram, "The Java Context-Awareness Framework (JCAF) – A service infrastructure and programming framework for context-aware applications.", *Pervasive*, 98–115, 2004.

[58] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, "Context-awareness on mo-bile devices – the hydrogen approach.", *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, pp. 292–302, 2002.

[59] J. Q. Quyang, Ding B., M. W. Huai, X. S. Dian, "Componenet Based Context Model", *Proceedings of The Ninth Intl. Web-Age Information Management (WAIM'08)*, pp. 569-574, July 2008.

[60] A. Devaraju, S. Hoh, "Ontology based Context Modeling for User-Centered Context-aware Services Platform", *Proceedings of Intl. Symposium on Information Technology (ITSim 2008)*, pp. 1-7, August 2008.

[61] D. Nicklas, M. Grossmann, J. Minguez, M. Wieland, "Adding High-level Reasoning to Efficient Low-level Context Management: A Hybrid Approach", *Proceedings of Sixth Annual IEEE Intl. Conf. On Pervasive Computing and Communications (PerCom 2008)*, pp. 447-452, March 2008.

[62] T. Gu, H. K. Pung, D. Q. Zhang, "A Service-Oriented Middleware for Building Context-Aware Services", *Journal of Network and Computer Applications*, pp. 1-18, 2005.

[63] D. R. de Almeida, C. de Souza Baptista, F. G. de Andrade, "Using Ontologies in Context-Aware Applications", *Proceedings of 17th Int. Conf. On Database and Expert Systems Applications (DEXA'06)*, pp. 349-353, 2006.

[64] O. Brdiczka, P. Reignier, J. L. Crowley, "Automatic Development of an Abstract Context Model for an Intelligent Environment", *Proceedings of Third IEEE Intl. Conf. on Pervasive Computing and Communications Workshops (PerCom 2005 Workshops)*, pp. 35-39, March 2005.

[65] G. Wang, J. Jiang, M. Shi, "Modeling Contexts in Collaborative Environment: A New Approach", *Computer Supported Cooperative Work in Design III*, Lecture Notes in Computer Science, Vol. 4402, Springer-Verlag, pp. 23-32, 2007.

[66] H. Q. Ngo, A. Shehzad, S. L. Kiani, M. Riaz, S. Y. Lee, "Developing Context-Aware Ubiquitous Computing Systems with a Unified Middleware Framework.", *Proceedings of Embedded and Ubiquitous Computing (EUC 2004)*, LNCS Volume 3207, Springer-Verlag, pp. 672 – 681, 2004.

[67] O. Cakmakci, J. Coutaz, K. Van Laerhoven, H. W. Gellersen, "Context Awareness in Systems with Limited Resources", *Proceedings of the Third workshop on Artificial Intelligence in Mobile Systems (AIMS)*, pp. 21-29, Lyon, France, 2002.

[68] R. Studer, V. R. Benjamins, D. Fensel, "Knowledge Engineering: Principles and Methods", *IEEE Transactions on Data and Knowledge Engineering,* 25(1-2), pp. 161-197, 1998.

[69] X. H. Wang, Q. D. Zhang, T. Gu, H. K. Pung, "Ontology Based Context Modeling and Reasoning using OWL", *Proceedings of Intl. Conf. Pervasive Computing and Communications (PerCom 2004)*, Orlando, FL, USA, 2004.

[70] A. Held, "Modeling of Context Information for Pervasive Computing Applications", *Proceedings of SCI2002, Orlando, Florida*, 2002.

[71] A. Gómez-Pérez, M. Fernández-López, O. Corcho, *Ontological Engineering*, Springer-Verlag, 2003.

[72] M. Dean, G. Schreiber, "OWL Web Ontology Language Reference", http://www.w3.org/TR/owl-ref/, 2004.

[73] X. Lin, S. Li, Z. Yang, W. Shi, "Application Oriented Context-Modeling and Reasoning in Pervasive Computing", *Proceedings of The Fifth International Conference on Computer and Information Technology*, 2005.

[74] E. Knutov, P. De Bra, M. Pechenizkiy, "AH 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques", *New Review of Hypermedia and Multimedia*, Vol. 15, No. 1, pp. 5-38(34), 2009.

[75] J. Mankoff, D. A. Gregory, S. E. Hudson, "OOPS: A toolkit supporting mediation techniques for resolving ambiguity in recognition-based interfaces", *Computers and Graphics*, 24(6), pp. 819-834, 2000.

[76] A. K. Dey, R. Hamid, C. Beckmann, I. Li , D. Hsu, "aCAPpella: programming by demonstration of context-aware applications", *Proceedings of Human Factors in Computing Systems (CHI 04)*, pp. 33-40, 2004.

[77] N. Hönle, Nicklas, U. P. Käppeler, T. Schwarz, M. Grossmann, "Benefits of integrating meta data into a context model", *Proceedings of the Third IEEE Conference on Pervasive Computing and Communications Workshops – Workshop on Context Modeling and Reasoning (CoMoRea)*, Kauai Island, Hawaii, USA, 2005.

[78] B. A. Truong, Y. K. Lee, S. Y. Lee, "Modeling and Reasoning about Uncertainty in Context-Aware Systems", *IEEE International Conference on e-Business Engineering*, pp. 102-109, 2005.

[79] T. Buchholz, A. Küpper, M. Schiffers, “Quality of context: What it is and why we need it”, *Proceedings of 10th International Workshop of the HP OpenView University Association (HPOVUA2003)*, Geneva, Switzerland, 2003.

[80] Y. Bu, J. Li, S. Chen, X. Tao, J. Lu, “An enhanced ontology based context model and fusion mechanism”, *Proceedings of IFIP 2005 International Conference on Embedded and Ubiquitous Computing (EUC2005)*, Nagasaki, Japan, LNCS Volume 3824, Springer-Verlag, pp. 920–929, 2005.

[81] A. Schmidt, “Ontology-based user context management: The challenges of dynamics and imperfection”, *Proceedings of ODBASE 2006*, On the Move Federated Conferences (OTM), Montpellier, France. LNCS, Springer, 2006.

[82] D. Ejigu, M. Scuturici, L. Brunie, “Semantic Approach to Context Management and Reasoning in Ubiquitous Context-Aware Systems”, *Digital Information Management*, 2007.

[83] G. Specht, T. Weithoner, “Context-aware processing of ontologies in mobile environments”, *Proceedings of the 7th International Conference on Mobile Data Management*, Washington, DC, USA, p. 86, IEEE Computer Society, 2006.

[84] A. Agostini, C. Bettini, D. Riboni, “A performance evaluation of ontology-based context reasoning,” *Proceedings of the 5th Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom’07)*, pp. 3–8,White Plains, NY, USA, March 2007.

[85] S. B. Mokhtar, D. Preuveneers, N. Georgantas, V. Issarny, Y. Berbers, “EASY: Efficient SemAntic Service DiscoverY in Pervasive Computing Environments with QoS and Context Support”, *Journal Of System and Software*, 81, pp. 785-808, 2008.

[86] J. Berri, R. Benlamri, Y. Atif, “Ontology Based Framework for Context-aware Mobile Learning”, *Proceedings of IWCMC’06*, Vancouver, Canada, 2006.

[87] A. L. Rector, “Modularisation of domain ontologies implemented in description logics and related formalisms including OWL”, *Proceedings of Second International Conference on Knowledge Capture (K-CAP)*, Sanibel Island, FL, 2003.

[88] D. Ejigu, M. Scuturici, L. Brunie, “An Ontology-Based Approach to Context Modeling and Reasoning in Pervasive Computing”, *Proceedings of CoMoRea Workshop of the IEEE International Conference (PerCom'07)*, New York, USA, 2007.

[89] A. Y. Turhan, T. Springer, M. Berger, “Pushing doors for modelling contexts with owl dl a case study”, *Proceedings of the Fourth IEEE Intl. Conf. on Pervasive Computing and Communications, Workshop on Context Modelling and Reasoning (CoMoRea’06)*, pp. 13–17, 2006.

[90] G. Biegel, V. Cahill, “A framework for developing mobile, context-aware applications”, *Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communication*, 2004.

[91] O. Kwon, J. Sim, M. Lee, “OWL-DL Based Ontology Inference Engine Assessment for Context-Aware Services”, *Agent and Multi-Agent Systems: Technologies and Applications*, Lecture Notes in Computer Science, Vol. 44-96, Spriger-Verlag, pp. 338-347, 2007.

[92] I. Roussaki, M. Strimpakou, N. Kalatzis, M. Anagnostou, C. Pils, “Hybrid context modeling: A location-based scheme using ontologies”, *Proceedings of 4th IEEE Conf. on Pervasive Computing and Communications Workshops*, pp. 2–7, 2006.

[93] G. Judd, P. Steenkiste, “Providing contextual information to pervasive computing applications”, *Proceedings of 1st IEEE Conference on Pervasive Computing and Communica-tions (PerCom)*, pp. 133-142, March 2003.

[94] B. Simon, D. Massart, V. F. Assche, S. Ternier, E. Duval, S. Branter, D. Olmedilla, Z. Miklos, “A Simple Query Interface for Interoperable Learning Resources”, *Proceedings of 1st Workshop on Interoperability of Web-based Educational Systems*, Chiba, Japan, 2005.

[95] K. Henricksen, J. Indulska, “A Software Engineering Framework for Context-Aware Pervasive Computing”, *Proceedings of Second IEEE International Conference on Pervasive Computing and Communications,* IEEE Computer Society, pp. 77–86, 2004.

[96] C. S. Hong, H. Kim, H. S. Kim, H. C. Lee, “An Approach for Configuring Ontology-based Application Context Model,” *Proceedings of IEEE International Conference on Pervasive Services*, Lyon, France, 2006.

[97] S. Helal, "Programming pervasive spaces," *IEEE Pervasive Computing*, Vol. 4, No. 1, pp. 84-87, 2005.

[98] M. Khedr, A. Karmouch “ACAI: Agent-Based Context-aware Infrastructure for Spontaneous Applications”, *Journal of Network and Computer Applications*, 28(1), pp. 19-44, 2005.

[99] H. Chen, T. Finin, A. Joshi, and L. Kagal, “Intelligent Agents Meet the Semantic Web in Smart Spaces”, *IEEE Internet Computing*, 8(6), pp. 69-79, 2004.

[100] C. Atkinson, T. Kühne, "Model-Driven Development: A Metamodeling Foundation", *IEEE Software*, September 2003.

[101] S. J. Mellor, A. N. Clark, and T. Futagami, “Model-driven development: Guest editor’s introduction”, *IEEE Software*, 20(5), 2003.

[102] G. Banavar, J. Beck, E. Gluzberg, J. Munson, J. Sussman, D. Zukowski, “Challenges: an application model for pervasive computing”, *Proceedings of 6th ACM MobiCom*, Boston, MA, USA, 2000.

[103] T. O. Meservy, K. D. Fenstermacher, “Transforming software development: An MDA road map”, *IEEE Computer,* 38(9) , pp. 52–58, 2005.

[104] A. Ranganathan, R. E. McGrath, R. H. Campbell, M. D. Mickunas, “Ontologies in a pervasive computing environment”, *Proceedings of the Workshop on Ontologies and Distributed Systems (part of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003),* Acapulco, Mexico, August 2003.

[105] B. Selic, “The Pragmatics of Model-Driven Development”, *IEEE Software*, 19–25. Special Issue on Model-Driven Development, 2003.

[106] M. H. Coen, “Design principles for intelligent environments”, *Proceedings of AAAI/IAAI 1998*, pp. 547–554, 1998.

[107] T. Kindberg and J. Barton, “A web-based nomadic computing system”, *Computer Networks*, 35(4), pp. 443–456, 2001.

[108] H. Chen, T. Finin, A. Joshi, “The SOUPA ontology for pervasive computing”, *Ontologies for agents: Theory and experiences*, Whitestein series in software agent technologies, Springer, 2005.

[109] M. Gruninger, J. Lee, “Ontology Applications and Design”, *Communications of the ACM*, 45(2), pp. 39–41, 2002.

[110] M. Uschold, M. Gruninger, “Ontologies: Principles, Methods and Applications”, *Knowledge Engineering Review*, 11(2), pp. 93–15, 1996.

[111] M. Uschold, R. Jasper, “A Framework for Understanding and Classifying Ontology Applications”,

Proceedings of IJCAI Workshop on Ontologies and Problem-Solving Methods, August 1999.
[112] N. Guarino, “Formal Ontology in Information Systems”, *Proceedings of FOIS’98*, Trento, Italy, IOS Press, Amsterdam, 1998.
[113] N. M. Goldman, “Ontology-oriented programming: static typing for the inconsistent programmer”, *Proceedings of International Semantic Web Conference (ISWC’2003)*, LNCS Vol. 2870, Sprinter-Verlag, Berlin, pp. 850–865, 2003.
[114] F. Ruiz, and J. R. Hilera, “Using Ontologies in Software Engineering and Technology”, *Ontologies for Software Engineering and Software Technology*, C. Calero, F. Ruiz and M. Piattini (Eds.), Springer Berlin Heidelberg, 2006.
[115] D. S. Frankel, “An MDA Manifesto”, *MDA Journal*, 2004.
[116] C. Anagnostopoulos, A. Tsounis, S. Hadjiefthymiades, “Context management in pervasive computing environments”, *Proceedings of Intl. Conf. on Pervasive Services (ICPS '05)*, 2005.
[117] A. Soylu, P. De Causmaecker, “Merging Model Driven and Ontology Driven System Development Approaches: Pervasive Computing Perspective”, *Proceedings of 24th International Symposium on Computer and Information Sciences (ISCIS’09)*, Guzelyurt, Northern Cyprus, 2009.
[118] N. F. Noy, D. L. McGuinness, “Ontology development 101: A guide to creating your first ontology”, Technical Report SMI-2001-0880, *Stanford Medical Informatics*, 2001.
[119] A. Eberhart, “Automatic Generation of Java/SQL based Inference Engines from RDF Schema and RuleML”, *Proceedings of the First International Semantic Web Conference (ISWC 2002)*, Chia, Sardinia, Italy, pp. 102--116 , June 2002.
[120] A. Kalyanpur, D. Pastor, S. Battle, J. Padget. “Automatic mapping of OWL ontologies into Java”, *Proceedings of the International Conference on Software Engineering & Knowledge Engineering (SEKE)*, 2004.
[121] I. N. Athanasiadis, F. Villa, A. E. Rizzoli, “Ontologies, JavaBeans and Relational Databases for enabling semantic programming”, *Proceedings of the 31th IEEE Annual International Computer Software and Applications Conference (COMPSAC)*, Vol 2, pp. 341-346, 2007.
[122] I. Astrova, N. Korda, A. Kalija, “Storing OWL Ontologies in SQL Relational Databases”, *International Journal of Electrical, Computer and Systems Engineering*, Vol. 1, No. 4, 2007.
[123] A. Soylu, P. de Causmaecker, “Embedded Semantics Empowering Context-Aware Pervasive Computing Environments”, *Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*, Brisbane, Australia, 2009.
[124] P. Debaty, P. Goddi, A. Vorbau, “Integrating the physical world with the web to enable context-enhanced mobile services”, *Mobile Networks Applications*, Vol. 10, No. 4, pp. 385-94, 2005.
[125] K. Sakamura, N. Koshizuka, “Ubiqutious computing technologies for ubiqutious learning”, *Proceedings of Intl. Workshop on Wireless and Mobile Technologies on Education*, Japan, 2005.
[126] M. Wilson, B. Matthews, “The Semantic Web: Prospect and Challenges”, *Proceedings of Database and Information Systems, 7th International Baltic Conferences*, pp. 26-29, 2006.
[127] B. Adida, M. Birbeck, “W3C RDFa Primer”, Available at http://www.w3.org/TR/xhtmlrdfa-primer/#id85078, 2008.
[128] Talis, “RDF in HTML: Embedded RDF”, Available at http://research.talis.com/2005/erdf/wiki/Main/RdfInHtml 2006.
[129] J. Simpson, “Microformats vs. RDF: How Microformats relate to the Semantic Web”, Available at http://www.semanticfocus.com/blog/entry/title/microformats-vs-rdfhowmicroformats-relate-to-the-semantic-web, 2007.

Ahmet Soylu is a PhD candidate in Katholieke Universiteit Leuven, Belgium. He received his BS degree and MSc degree in computer engineering and science from Işık University, Istanbul, Turkey in 2006 and 2008 respectively.

He worked as a full time Researcher in Informatics Research and Development Centre of Işık University during his master studies. Currently he is a part of iTec and CODeS research groups, Kortrijk, Belgium. His research interests include, e-learning, pervasive and context-aware computing, software engineering and database systems.

Patrick De Causmaecker received MSc degree in mathematics from University of Ghent, Belgium in 1978. He obtained his PhD degree in theoretical physics from Katholieke Universiteit Leuven, Belgium in 1983. Later he received a MSc degree in computer science from University of Ghent, Belgium in 1987.

He is currently a Professor of Computer Science at the Subfaculty of Sciences at Katholieke Universiteit Leuven, Campus Kortrijk, Belgium. He is the head of the CODeS Research group in the department of Computer Science and coordinator of the iTec research group, Kortrijk, Belgium. His current research interests include optimization, planning and scheduling, distributed scheduling applications, agent technologies, and pervasive computing.

Piet Desmet received MA degree in Romance Languages and Literatures from the K.U. Leuven, Belgium in 1987. He obtained his PhD on French linguistics from the K.U. Leuven, Belgium in 1994.

He is full Professor of French and Applied Linguistics at the Faculty of Arts at K.U. Leuven and its campus Kortrijk. He is co-directing iTec research group (www.itec-research.eu). His current research interests include computer-assisted language learning, French language teaching, and educational technology.