# CANDEL: Product Line Based Dynamic Context Management for Pervasive Applications

Zakwan Jaroucheh
School of Computing
Edinburgh Napier University
Scotland, UK
z.jaroucheh@napier.ac.uk

Xiaodong Liu
School of Computing
Edinburgh Napier University
Scotland, UK
x.liu@napier.ac.uk

Sally Smith
School of Computing
Edinburgh Napier University
Scotland, UK
s.smith@napier.ac.uk

*Abstract*— **In pervasive environment, it is essential for computing applications to be context-aware. However, one of the major challenges is the establishment of a generic and dynamic context model. Many different approaches to modeling the context exist, but an application- and domain-agnostic context model, that captures various types of context information and dependency between them, that could be reused and shared by different applications, and that can be dynamically changed when a shift in focus occurs, is missing. Therefore, we are interested in defining a structure for the dynamic management of context information. This paper describes our notion of context and proposes distributed context management architecture that supports the development of context-aware applications. It presents CANDEL, a generic context information representation framework that considers the context as a dynamic product line composed of context primitives (CPs). Frame based software product line techniques are used together with OWL ontology to define CPs and to dynamically generate the current context model. Further, using Petri-Nets, we also show how this framework will be used to support the context-aware adaptive pervasive applications.**

***Keywords-** ontology-based context model; pervasive applications; software product line; feature model.*

## I. INTRODUCTION

Pervasive computing introduced the concept of anytime and anywhere computing. It includes a wide variety of devices (e.g., mobile phones, PDAs) that are becoming continuously present in our daily tasks. In this new paradigm, the focus now is to amplify human activities with new services that can be adapted to the circumstances in which they are used [1].

Context-aware systems are part of this dynamic scenario. These systems use context information to provide relevant services and information for their users. To ease the development of such applications it is necessary to decouple application from context acquisition and representation, and at the same time provide universal models and mechanisms to manage context. Thus, generic and dynamically manageable context models are of interest since they can be reused by different applications and ease context sharing between systems [2].

The aim of CANDEL (Context As dyNamic proDuct Line) is to address the above mentioned challenges to manage the context information in a domain-independent way, with particular emphasis on the notion of context. As part of our approach it is important to build the context manager upon a generic context model.

As one of the successful research directions in software engineering, software product line research could contribute to the context modeling. Commonality and variability management techniques from software product line can be applied to handle context variabilities for customization and adaptation. Therefore, in this paper we explore the synergy between feature modeling and context modeling.

The key idea applied in CANDEL is to separate the context ontology management among different components called Context Proxy Components (CPCs); and to apply the software product line idea in each CPC to dynamically get a customized part of the context information it manages. We propose to look at the whole system context as multiple product lines supporting several dimensions in the context space.

On the other hand, feature modeling is a key concept in product line engineering. Thus, the feature model of the system context will be considered as a composition of segmented context features models; each of which models a part of the whole context. Based on the context feature model, specific context −member of a product line− can be constructed by composing features from context information.

The paper is organized as follows: the context-awareness concept and a new working context definition are presented in Section II. In Section III, context modeling requirements are presented. Section IV is a brief introduction to Software Product Line. In Section V, we present CANDEL, our conceptual framework, and describe the context management architecture. The context model as a dynamic product line idea is introduced in Section VI. In Section VII we show how CANDEL support pervasive applications using Petri-Nets. Section VIII briefly report on other context models, considering their relevant features and comparing them with CANDEL, and finally the conclusions are drawn.

## II. CONTEXT-SENSITIVE APPLICATIONS

Context-sensitive applications are those that consider the current situation of their users in order to provide services and information tailored to their needs. An important topic when dealing with context-aware systems is how to model, manage, and manipulate the context information. To ease

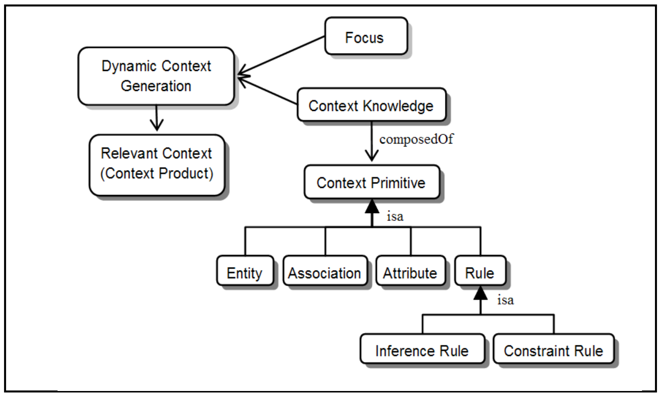


Figure 1. The proposed context working definition.

context representation, context sharing and semantic interoperability between heterogeneous systems, a formal and generic context model is needed [3].

In this work, we are interested in developing a context-aware application development methodology (Software Engineering perspective); and in particular we are focusing on context modeling (Knowledge Engineering perspective).

In the literature, there are many definitions for context. Definitions given by earlier works agree on the key idea that contexts describe situations. For example Dey [4] confirmed this by defining context as: "Any information that can be used to characterize the situation of an entity. An entity is a person, a place, or a physical or computational object that is considered relevant to the interaction between a user and an application, including the user and application themselves."

This work is based on two other definitions of context. The first states that in using open-ended phrases such "any information" and "characterize" the context becomes so broad that it covers everything [5]. Winograd [5] indicated also that "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." In this work, we adopt his definition of context: "context depends on the interpretation of the operations involved on an entity at a particular time and space rather than the inherent characteristics of the entity itself."

The second indicates that "context is always related to a focus and that, at a given focus, the context is the aggregation of three types of knowledge: Contextual Knowledge (CK), External Knowledge (EK) and Proceduralized Context (PC)" [6]. The authors in [6] argue that context should always be considered related to a focus, which is a step in a task execution, in a problem solving or in a decision making process. Moreover, the context evolves dynamically according to the focus, which enables a context-sensitive system to separate relevant from not relevant knowledge in order to determine the context.

Fig. 1 illustrates the proposed working definition of the context. The term context primitive (for short, we will refer to it as CP) refers to a piece of contextual knowledge such as entity, entity attribute, relationship between two entities, their constraints, or inference rules –used to define context situations and infer new knowledge– that can be used to define the context. We consider that the context knowledge is composed of a set of small pieces. Given a focus, a relevant subset of these pieces, namely context primitives, will be used to generate the current context. Thus, the generated context is in alignment with the requirement of current task.

Several authors mentioned the distinction between data, information and knowledge e.g. [7]. The raw data "Alice is located in Kitchen" is represented by composing the primitives: Entity (Alice), Association (is located in), and Entity (Kitchen). In the same way we represent "Alice is a female" as a composition of: Entity (Alice), Attribute (Gender), DataValue (Female), and Constraints (Male or Female). Information is a relationship between data with great dependence on context for its meaning [7]. To understand the relationship between these raw data and therefore to conclude meaningful information, these data should be associated to a context. Here, we consider the focus is the context under which the data could be understood and interpreted. For instance, if the focus was to know the activity of Alice then we may conclude that Alice is cooking; in this case we use the Rule primitive to infer the new information "Alice is cooking". In contrast, if the focus was just to know the position of Alice then we use the Rule primitive to transform the data representing the coordinates of Alice into a meaningful information e.g. in kitchen or bathroom.

Beyond relation between data there is pattern which has the potential to represent knowledge. In this respect, based on the information that "Alice cooks everyday", we use the Rule primitives to conclude the knowledge that, for example, Alice is a housekeeper or Alice likes cooking. In brief, the focus determines what are the context primitives to be considered when dynamically compose the current context.

## III. Context Modeling Requirements

A context model is needed to define and store context data in a machine processable form. Ontologies are a very promising instrument for modeling contextual information due to their high and formal expressiveness and the possibilities for applying ontology reasoning techniques [8]. Thus, ontologies will be used in the context model as the underlying technology.

The development of the proposed context modeling approach was driven by requirements we collected from the literature and from our experience in the context management implementation in pervasive environment. Besides the context modeling requirements mentioned in [9]: applicability, comparability, traceability, history and logging, quality, inference, incompleteness and ambiguity, we also identify the following requirements when designing an ontology-based context model:

**R1- Context modeling should provide applications with customized subset of the context information**. A complete ontology-based context model contains every piece of knowledge to be used for all application scenarios, which represents all variants in the domain. In reality, not all context information is needed in all application scenarios [1]. Therefore, an appropriate subset of the complete context knowledge for a specific situation is enough.

**R2- The context modeling should provide consistency checking mechanism**. The context model should define the range a context value can take, or define a particular co-existence of values to be impossible; and thus should provide mechanisms to check the consistency of the context information instances.

**R3- A generic approach to context modeling is needed**. Existing context models vary in context information they can represent. While some models take the user situation into account, others model the physical environment. Thus, a more general approach that captures various types of context information, dependency between them and the context history is needed [2, 10], so that context models can be reused by different applications and ease context sharing between systems.

**R4- Context model should provide context information in different level of abstraction**. It should hide irrelevant context details and [1] offer a high-level interpretation of lower-level context details if requested.

**R5- Context modeling should be domain and application-agnostic**. The reusability criterion requires that the context representation should be independent from application. Instead the application is expected to be context-dependent. Further, context modeling concepts should be independent of the domain.

**R6- Context modeling should rely on well-accepted standards for expressing context information**. This will guarantee the interoperability between devices and applications in the highly-dynamic pervasive environment.

**R7- Context modeling should have formal representation of its syntax and semantics** in order to guarantee the consistency between different representations of context used by applications, context providers and service platforms [15]. This may require formally defining a context metamodel that will be used to produce valid and consistent context model throughout the system development, deployment and operation.

**R8- The context model should integrate the quality of context information** as the quality of context information delivered by sensors change over time.

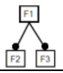
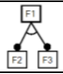
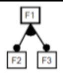
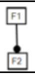
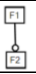
**R9- The context model should be dynamic**. In order to reflect the underlying dynamic nature of the pervasive environment, the context model should cater for addition and removal of context data sources and address dynamic value changes of these sources. Further, as will be seen later, it should change when a shift in focus occurs.

In addition, we identify the context management framework requirements:

**R10- It should be reusable**. The context manger should be lightweight in terms of computational resources and interfacing requirements to be reused in as many environments as possible.

**R11- It should provide infrastructure to facilitate the construction, deployment and execution of context-aware applications.** Particularly, support is needed to handle different sources and types of contextual information, provided by highly distributed heterogeneous and constantly changing environments.

TABLE I. FEATURE TYPE RELATIONS

| | |
|---|---|
| **And**: if F1 is selected, subfeatures (F2,F3) must be part of any product of the product line |  |
| **Alternative**: if F1 is selected, only one subfeature (F2 or F3) can be selected in any product in the product line. |  |
| **Or**: if F1 is selected, one or more subfeatures can be selected as part of any product in the product line. |  |
| **Mandatory**: if F1 is selected, the subfeature is required as part of any product in the product line. |  |
| **Optional**: if F1 is selected, the subfeature may or may not be part of a product in the product line. |  |

## IV. SOFTWARE PRODUCT LINE

According to [11] a software product line (SPL) is a set of software-intensive systems sharing a common, managed set of features that satisfy specific needs of a particular market or mission, and that are developed from a common set of core assets in a prescribed way.

Feature modeling is a domain modeling technique, which has generated a lot of interest in the software product line (SPL) community [12]. Modeling product family as a hierarchy of features their similarities, differences and relationships among them, feature models can be used for modeling common and variable requirements of products in a SPL, scoping SPLs, and product configuration and derivation.

Commonly there are five types of relations possible in a feature model [13] (See Table 1). Additional constraints between features may exist that describe how features interact with each other e.g. requires and excludes constraints.

In the following section we explain how we can benefit from SPL techniques in the context modeling and management.

## V. CONTEXT MODEL CONCEPTUAL FRAMEWORK

Based on the requirements mentioned in Section III, and inspired by the idea that feature models are views on ontologies [14], we propose a SPL-based conceptual metamodel to support context-aware application development. We feel that there is a strong similarity between feature modeling and context modeling, both of which represent concepts in a particular domain and define how various properties relate among them. Hence, similarly to feature model, the context is presented as a feature diagram with other associated information such as constraints and dependency rules. On the other hand, since the context information may come from multiple heterogeneous sources, it is important to think of formalism and common languages that enable context sharing and interoperability of these sources in different applications [6].

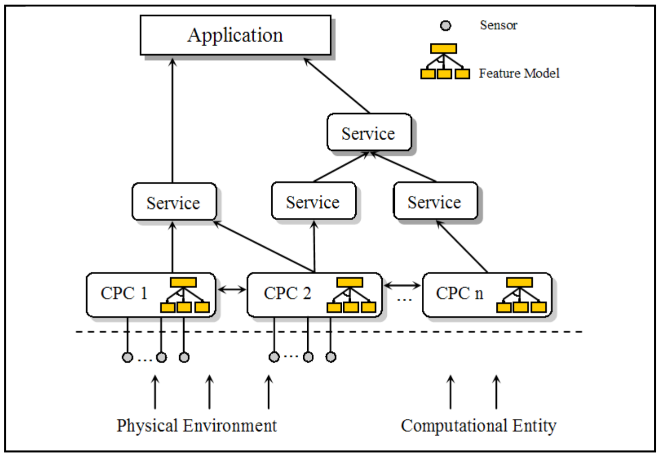


Figure 2. Context Management Architecture

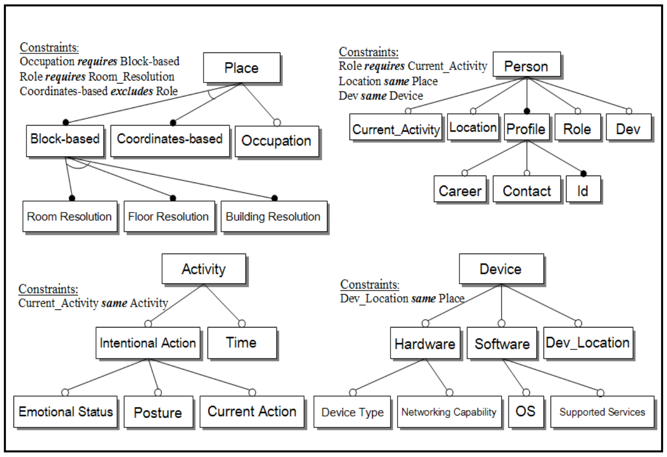


Figure 3. Context Feature Models

Hence, we are interested in investigating mechanisms for context modeling in a generic manner to support different applications in a domain-independent way.

### *A. Motivation Scenario*

Alice registers her preferences when booking a room in the City Hotel. Once she gets into her room, the existent services use her context and preferences to deliver the preferable ambiance (e.g. light level, music type and level, etc). She is interested in reading IT news at night. The news service is willing to help her in reading the most recent IT news. After entering the room, she may take her shower. In this case, the service presumes that she is in “awaked mode” so it delivers the news to her mobile device. After a while, she may become sleepy, and the service delivers the news to the LCD screen or provides it in an auditory form. But if after entering the room, she had her dinner, and it was night, this means that she is in “sleepy mode”, and then the TV switches to his favorite relaxing music channel and the lights dim.

In the following sections we present the proposed conceptual framework that supports this scenario.

### *B. Context Management Architecture*

In the proposed approach we consider that context management is embedded in entities, called Context Proxy Components (CPCs) (Fig. 2), distributed in the environment, and publishes its capacity in the form of feature model to a service directory so that they can be easily located. These entities include any device or computational entity e.g. service. We have two kinds of these proxies: proxies connected to sensors that aggregate their data to provide services with the physical environment context, and proxies that provide the non-physical context e.g. computational entities.

For instance, Fig. 3 shows different context feature models provided by four CPCs. In Fig. 3, the Place CPC can provide the location as two features: the coordinates-based (i.e., longitude and latitude) or block-based which in turn has three subfeatures: room-, floor- or building-resolution. The service interested in knowing in which room the person is located should select the Location feature from the Person CPC. As the Location is the same as Place (see next section), the service has to select the subfeatures: Block-based and Room-resolution.

### *C. Context Model Interoperability*

As other works (e.g. [2, 12]) have mentioned, ontologies are appropriate tools for representing context information. Ontologies are thus used to define context elements. On the other hand, Semantic Web is very promising to enhance the knowledge sharing; it offers knowledge representation languages that are both expressive and open which are two useful features for expressing context. With context models expressed in OWL DL, a Semantic Web environment can be built to facilitate context storing, sharing and distribution and to assist design cooperation.

The CPCs maintain their own context expressed in OWL ontologies. They may communicate this information to other CPCs, obeying a simple interface for determining the information they can provide using the context feature model. Indeed, in pervasive environment the growth in terms of the number of CPCs calls for distributing the process of creating context feature models. Two forms of distribution can be identified; distribution due to the fact that different people may be involved in context feature model creation, and distribution due to defining different context feature model for different parts of the system. The context consumer service may communicate with different CPCs to get a “snapshot” of the context information it needs.

Further, because the context of each CPC cannot be considered in isolation from the other contexts, there may be many relationships (dependencies) between the features of different CPCs. For instance, in Fig. 3, the *Role* feature in

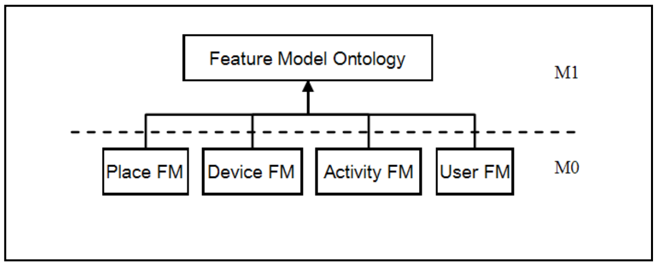


Figure 4. Feature Model Ontology Framework

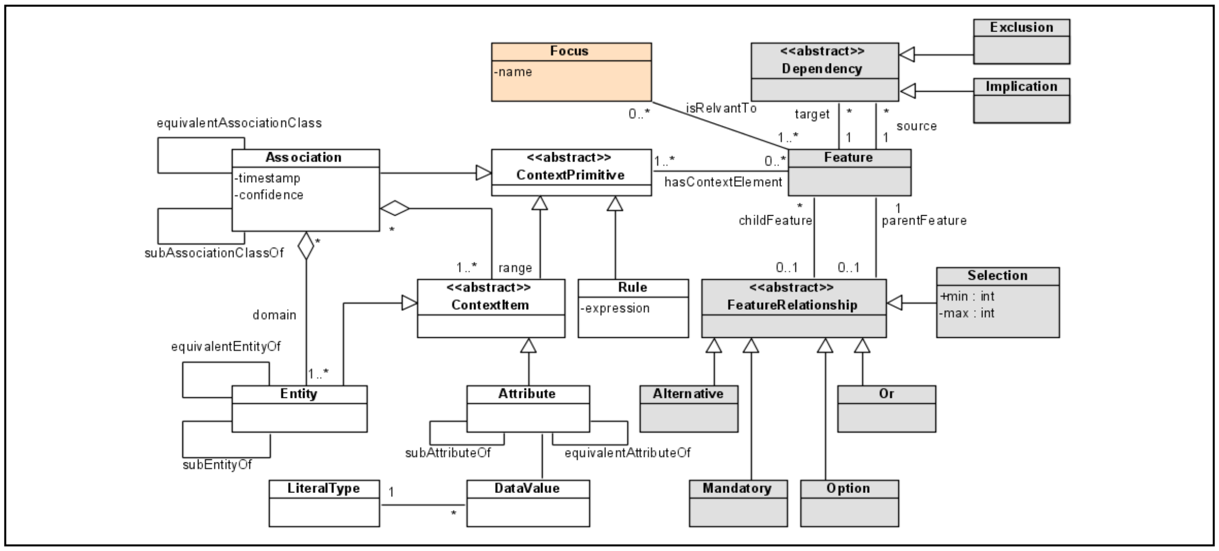


Figure 5. The Conceptual Meta-Model

*User FM* requires the *Current_Activity* feature in *Activity FM* (Role *requires* Current_Activity); which means that in order for the Person CPC to provide the role information it needs the current activity information from Activity CPC. Another type of constraint: the feature Location is the same as Place feature (Location *same* Place).

Therefore, features-based context modeling approach faces the challenge: lack of a formal common semantics for context feature models.

In order to be able to integrate the different context feature models, we employ ontology-based approach for the representation of knowledge contained in these models. As will be seen later, we use ontology to describe the feature model that will be used as a *meta-model* for describing different features models of CPCs (Fig. 4). Thus, every context feature model is an instance of the feature model ontology.

Further, to facilitate the development of context-aware application, it is necessary that applications and supporting platforms share not only a common feature metamodel but also a common context metamodel. Therefore, we propose to integrate the context feature model and context model in the meta level. The concepts of the conceptual metamodel were identified and grouped into two different views (Fig. 5): the context related concepts (white), and the context features concepts (shaded).

We import the concepts of features from FODA (Feature Oriented Domain Analysis) [15]. FODA appeals to us because features are essential abstractions that both context consumer and provider understand. Thus, the main concept in the feature description language FODA is the feature itself. Here a feature is a set of context primitives that is relevant to some stakeholder from a specific “focus” point of view. Fig. 5 depicts the proposed conceptual metamodel.

The main construct for representing context knowledge is the ContextPrimitive which represents the base context constructs (primitives) mentioned above: entity classes, entity attributes, entities associations, and rules.

- Entity class: represents a group of entities (e.g. users, places, devices, etc) sharing some properties.
- Attribute class: represents entities attributes e.g. position, temperature, etc.
- Association class: represents a relationship between one entity and either another entity or an attribute.
- Rule class: two types of rules could be identified: (i) *Consistency rules* provide mechanism for context consistency by specifying conditions that must be hold in the context information. For example, consistency rule could specify that if the person is cooking, she must be in the kitchen. (ii) *Inference rules* used to generate new context information after reasoning on the existing one. For example, an inference rule could conclude that a person is sleeping if the light is off and the time is night.

Further modeling constructs are axioms that add additional facts about the entities and attributes. These are: *specialization* and *equivalence* relationships that may be specified between two entity classes, two attribute classes, or two association classes.

## D. *Modeling Constructs on the Metamodel Layer*

For the formal specification of the conceptual metamodel, we have leveraged the existing ontology language OWL DL [16] to represent the knowledge contained in the conceptual model for a number of reasons. First, in OWL (and description logics), conceptual entities are organized as classes in hierarchies. Individual entities are grouped under classes and are called instances of the classes. Classes and individuals can be related by properties. OWL has constructs to define set relations including subclass, equivalence, intersection, union, etc. This facilitates the transition from the proposed conceptual view of the model to the ontological view of the model. Second, OWL is the W3C standard for Semantic Web which eases the exchange of context models between context consumer and providers. Third, the reasoning capabilities of OWL DL are of crucial importance to context-aware applications for context knowledge representation and reasoning. The Description Language (DL) reasoners are used, on one hand to infer knowledge using rules implemented in the Semantic Web Rule Language SWRL [17], and on the second hand to ensure model consistency.

In the following we briefly explain the OWL constructs used for ontology-based conceptual model representation.

**a) Conceptual Model Ontology Class Constructs**

We have defined the following class constructs:

- *ContextPrimitive* is the main ontology construct which is the super-class of other context constructs.
- *Entity, Attribute, Association, DataValue, LiteralType*: are OWL classes representing *EntityClass, AttributeClass, AssociationClass, DataValueClass, and LiteralType* respectively.
- *Feature*: is the main ontology construct representing the context Feature.
- *Focus*: represents the Focus concept.

**b) Conceptual Model Ontology Property Constructs**

- *Feature_to_Feature_Relationship* (FFR) property which has the Feature class as both domain and range.
- *Feature_to_Feature_Dependency* (FFD) property which has the Feature class as both domain and range.
- *isRelevantTo* object property corresponds to *isRelevantTo* property. It has the Focus class as a domain and Feature class as range.
- *hasContextPrimitive* object property corresponds to *hasContextPrimitive* property. It has the Feature class as a domain and *ContextPrimitive* class as range.
- *hasDomainEntity* and *hasRangeEntity* object properties have the *AssociationClass* as domain and Entity class as range, which represent the relationships between entities.
- *hasRangeAttribute* object property has the *AssociationClass* as domain and *AttributeClass* as range, which represents the entity’s attribute.
- *timestamp* datatype property represents the time of assigning a relationship between two entities or assigning an attributes value to an entity. It has an *AssociationClass* as domain. It can be used when considering the context history.

**c) Conceptual Model Ontology Axioms**

We define also *And, Alternative, Mandatory, Or, Option,* and *Selection* as sub-properties of FFR. We define also *Requires* and *Excludes* as subPropertyOf FFD representing the *Implication* and *Exclusion* dependencies respectively.

The properties *Requires* and *Excludes* are defined as mutual exclusive properties.

*Specialization* and *Equivalence* relationships -mentioned above- that may be specified between two entity classes, two attribute classes, or two association classes, are realized as OWL subClassOf and equivalentClass axioms respectively.

**d) Conceptual Model Ontology Rules**

Here, we mean by rules the derivative rules define by the system developer in order to infer context knowledge or define context situations. We implement the rules described by the metaclass Rule in the proposed conceptual model in SWRL. SWRL is an expressive OWL-based rule language. SWRL allows users to write rules that can be expressed in terms of OWL concepts to provide more powerful deductive reasoning capabilities than OWL alone. The SWRL metamodel defines Rule as a subclass of OntologyElement [18]. Thus, we represent the metaclass Rule as a Rule class in SWRL metamodel.

## VI. CONTEXT AS A DYNAMIC PRODUCT LINE

As we have already mentioned the context evolves dynamically according to the focus and that context is a set of contextual elements that are assembled and instantiated according to the focus. In this section, due to space limitation, we briefly explain how the context manager can build the context model given a set of features.

The idea is as follows: starting from the context model, we manage the underlying context knowledge using the techniques of how commonalities and variabilities are handled in a product line. Using this knowledge we build a context product line, in which customized context product could be build given the context features different services are interested in.

One of the effective ways to deal with handling variants is to use the XVCL variability mechanism [19] that supports automated customization and assembly of product line assets. Using XVCL, we develop product line ontology assets (meta-ontologies) as a set of x-frames that incorporate both context defaults and variants. X-frames represent the context knowledge in the form of product line assets. Specific context, members of a product line, can be constructed by composing these meta-ontologies. More details about XVCL are in [19].

For each CPC, we develop the context product line architecture based on the context ontology architecture. Using XVCL, we design generic components as x-frames that incorporate both context defaults and variants. The resulting x-frames are meta-components (meta-ontologies), from which concrete components are constructed during the process of producing a specific context product using the reusable assets.

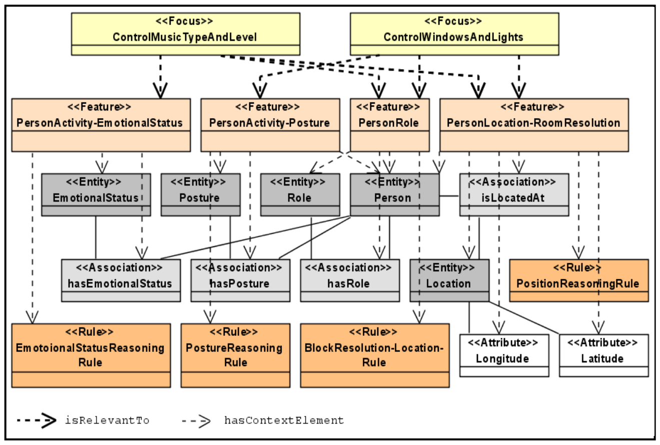

<<Focus>> ControlMusicTypeAndLevel
<<Focus>> ControlWindowsAndLights
<<Feature>> PersonActivity-EmotionalStatus
<<Feature>> PersonActivity-Posture
<<Feature>> PersonRole
<<Feature>> PersonLocation-RoomResolution
<<Entity>> EmotionalStatus
<<Entity>> Posture
<<Entity>> Role
<<Entity>> Person
<<Association>> isLocatedAt
<<Association>> hasEmotionalStatus
<<Association>> hasPosture
<<Association>> hasRole
<<Entity>> Location
<<Rule>> PositionReasoningRule
<<Rule>> EmotoionalStatusReasoning Rule
<<Rule>> PostureReasoning Rule
<<Rule>> BlockResolution-Location-Rule
<<Attribute>> Longitude
<<Attribute>> Latitude
isRelevantTo
hasContextElement

Figure 6. Association between the focus and the context primitives

## VII. A Petri-Net Based Approach for Context-Aware Adaptive Applications

Following Dey's context definition, *situation* is a central notion describing context. Dey [4] defines situation as a "description of the states of relevant entities". We call a specified set of contextual information acquired during one instance of time a *situation*; and a specified secession of situations is called a *behavior*.

Based on motivation scenario (Section V.A), Fig. 6 shows that different set of context features are selected by two different tasks (different focuses). Therefore, music control service selects context features different from those selected by the lights and windows controlling service.

To develop a context-aware adaptive applications, and to represent the relation between "focus" and different context-sensitive behaviors for a system's adaptation we use Predicate/Transition nets (PrT nets) [20], a kind of widely used high-level Petri nets, to model context situations and behaviors. The rationale behind this approach is that PrT nets are suitable for dealing with logic and rules as well as in dealing with temporal aspects. In particular, we propose using a special kind of PrT, the rule nets, to model the situations. The adaptation actions are triggered when the acquired context information corresponds to a specified situation or the history of context information corresponds to a specified behavior.

In this approach we benefit from expressiveness power of Petri Net to model the behavior; that is the succession of situations. According to the motivation scenario already mentioned, Fig. 7 (a) illustrates modeling two situations: the guest is taking rest in living room, and the guest is nervous while in bed room. In this respect, the application sends to the correspondent CPCs the feature list it is interested in. In this case, the application expresses its interest in the *Role*, *Room-resolution, and Posture* features from *Person*, *Location* and *Activity* CPCs respectively by sending the feature ids with the corresponding parameters via the CPC interface. Fig. 7 (b) illustrates modeling different scenarios of the guest behaviors. It illustrates how to model the transition between situations.

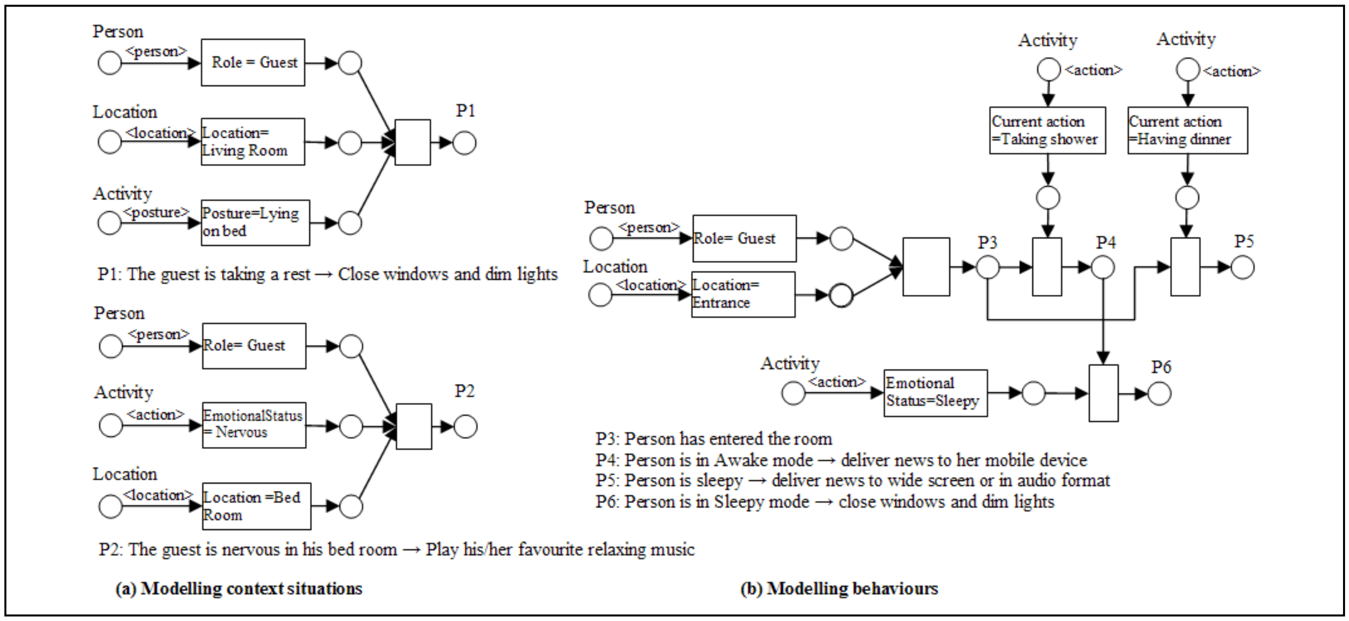
## VIII. Related Work

Different ontologies have been proposed in the literature to model domain specific context information (e.g. [2]) or generic models reusable in different domains (e.g. [10, 21]) but all with certain drawbacks in genericity and/or dynamicity.

CONON [21] is composed of an upper context ontology, which defines the basic concepts of context and must be extended by the developer to achieve a domain specific context model. The interest of these approaches is being able to extend ontologies. However, whenever one agree on using a particular ontology, the model is bound to the assertions made therein including the ones that potentially contradict the semantics to be modeled. In contrast, in CANDEL, system developers have the flexibility to specify the context and its structure according to the application/domain.

CoOL [22] is an ontology-based Aspect-Scale-Context (ASC) model that supports for interoperability. Each aspect aggregates one or more scales, and each scale aggregates one or more context information. The context model of CANDEL is more expressive than ASC model. The Feature concept can aggregate an arbitrary number of context primitives which are more generic than the context information. Further, the mapping restriction between Scales could be expressed in more generic way as the relationship and dependency between Features.

The Context Broker Architecture (CoBrA) is a broker-centric agent architecture that provides knowledge sharing, context reasoning, and privacy protection [2]. It is domain-specific and only covers contexts in campus space; it has no explicit support for modeling general contexts in heterogeneous environments.



Person
<person>
Role = Guest
Location
<location>
Location= Living Room
Activity
<posture>
Posture=Lying on bed
P1
P1: The guest is taking a rest → Close windows and dim lights
Person
<person>
Role= Guest
Activity
<action>
EmotionalStatus = Nervous
Location
<location>
Location =Bed Room
P2
P2: The guest is nervous in his bed room → Play his/her favourite relaxing music
(a) Modelling context situations
Activity
<action>
Current action =Taking shower
Activity
<action>
Current action =Having dinner
Person
<person>
Role= Guest
Location
<location>
Location= Entrance
P3
P4
P5
Activity
<action>
Emotional Status=Sleepy
P6
P3: Person has entered the room
P4: Person is in Awake mode → deliver news to her mobile device
P5: Person is sleepy → deliver news to wide screen or in audio format
P6: Person is in Sleepy mode → close windows and dim lights
(b) Modelling behaviours

Figure 7. Petri-Nets based context situations and behaviours modelling

In [23] the authors proposed a modelling technique for context information based on a modelling concept that embraces four abstraction layers from meta-metamodel and metamodel to model and instance layer. This enables the construction of restricted ontologies that comply with OWL DL and can be used to define context models. However, they did not mention how to dynamically generate a customized subset of the context information depending on the application needs nor did they mention how the application can acquire the context information in different levels of abstractions.

Context-Oriented Model approach [6] proposes the explicit separation of generic context concepts from specific concepts of an application domain. Their work is similar to ours; however, they did not consider the distribution aspect of the context management.

## IX. Conclusion and Future Work

In this paper we first presented a set of requirements that context modeling and context management should meet. A generic conceptual metamodel (CANDEL) was introduced. It provides a domain-agnostic formal representation of contextual information which promotes interoperability between applications e.g. context remembrance.

CANDEL integrates the context primitive (CP) specifications with the product line concepts that enable the dynamic context manipulation when focus changes. In CANDEL, based on software product line techniques, a specific context −member of a product line− can be dynamically constructed by composing features from context primitives. In fact, having different views of the same context information and at different levels of abstraction is one of the motivations in CANDEL to address the genericity issue. Furthermore, we discussed how the adaptation of the application to context is driven by the Petri net representation of the context situations and behaviors.

The development of the proposed metamodel was the first step towards a model-driven approach for the development of context-aware adaptive applications. OWL, XVCL, Java and Semantic Web technologies are used in implementing the context manager prototype and interfacing protocol between CPCs themselves and between CPCs and context consumers. Further, we aim to refine the proposed conceptual metamodel and extend it to support the self-adaptive process-oriented context-aware applications by introducing the change set primitives concepts.

## References

[1] T. Pederson, C. Ardito, P. Bottoni, and M. F. Costabile, "A General-purpose Context Modeling Architecture for Adaptive Mobile Services," presented at Proceedings of the ER 2008 Workshops (Cmlsa, Ecdm, Fp-Uml, M2as, Rigim, Secogis, Wism) on Advances in Conceptual Modeling: Challenges and Opportunities, Barcelona, Spain, 2008.

[2] H. Chen, T. Finin, and A. Joshi, "An Ontology for Context-Aware Pervasive Computing Environments," The Knowledge Engineering Review, vol. 18, pp. 197-207, 2004.

[3] T. Gu, H. Pung, and D. Q. Zhang, "A service-oriented middleware for building context-aware services," Journal of Network and Computer Applications, vol. 28, pp. 1-18, 2005.

[4] A. K. Dey, "Understanding and Using Context," Personal and Ubiquitous Computing, vol. 5, pp. 4-7, 2001.

[5] T. Winograd, "Architectures for Context," Human-Computer Interaction, vol. 16, pp. 401-419, 2001.

[6] V. Viera, P. Brézillon, A. C. Salgado, and P. Tedesco, "A Context-Oriented Model for Domain-Independent Context Management," Revue d'intelligence artificielle, vol. 22, pp. 609-627, 2008.

[7] G. Bellinger, "Knowledge Management - Emerging Perspectives," In: http://www.systemsthinking.org/kmgmt/kmgmt.htm, 2004, Access in August 2009.

[8] J. Euzenat, J. Pierson, and F. Ramparany, "Dynamic context management for pervasive applications," The Knowledge Engineering Review, vol. 23, pp. 21-49, 2008.

[9] R. Krummenacher, H. Lausen, T. Strang, and J. Kopecky, "Analyzing the Modeling of Context with Ontologies," presented at International Workshop on Context-Awareness for Self-Managing Systems (Devices, Applications and Networks) - CASEMANS 2007 - as part of Pervasive 2007, Toronto, Canada, 2007.

[10] T. Chaari, D. Ejigu, F. Laforest, and V.-M. Scuturici, "A comprehensive approach to model and use context for adapting applications in pervasive environments," Journal of Systems and Software, vol. 80, pp. 1973-1992, 2007.

[11] L. M. Northrop, "SEI's software product line tenets," IEEE Software, vol. 19, pp. 32-40, 2002.

[12] Z. Lamia Abo, K. Frederic, and T. Olga De, "Applying semantic web technology to feature modeling," in Proceedings of the 2009 ACM symposium on Applied Computing. Honolulu, Hawaii: ACM, 2009.

[13] H. H. Wang, Y. F. Li, J. Sun, H. Zhang, and J. Pan, "Verifying feature models using OWL," Web Semant., vol. 5, pp. 117-129, 2007.

[14] C. Krzysztof, K. Chang Hwan Peter, and K. Karl Trygve, "Feature Models are Views on Ontologies," in Proceedings of the 10th International on Software Product Line Conference: IEEE Computer Society, 2006.

[15] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," in Distribution. Pittsburgh, PA: CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 1990.

[16] C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler, and M. Smith, "OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax," 2008.

[17] "SWRL: A Semantic Web Rule Language Combining OWL and RuleML", W3C Member Submission 21 May 2004 <http://www.daml.org/2003/11/swrl/>.

[18] S. Brockmans and P. Haase, "A Metamodel and UML Profile for Rule-extended OWL DL Ontologies– A Complete Reference.," Lecture Notes in Computer Science, 2006.

[19] H. Zhang and S. Jarzabek, "An XVCL-based Approach to Software Product Line Development," presented at Int. Conf. on Software Engineering and Knowledge, SEKE’03, 2003.

[20] H. J. Genrich, "Predicate/Transition Nets," presented at Advances in Petri nets 1986.

[21] X. H. Wang, T. Gu, D. Q. Zhang, and H. K. Pung, "Ontology Based Context Modeling and Reasoning using OWL," presented at Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004.

[22] T. Strang, C. Linnhoff-Popien, and K. Frank, "CoOL: Context Ontology Language to enable Contextual Interoperability," Lecture Notes in Computer Science, vol. 2893, pp. 236-247, 2003.

[23] F. Fuchs, I. Hochstatter, M. Krause, and M. Berger, "A Metamodel Approach to Context Information" presented at Third IEEE International Conference on Pervasive Computing and Communications Workshops, 2005.