Journal of Software Engineering and Applications, 2014, 7, 311-346
Published Online April 2014 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2014.74030

Models, AmI-Creator and A-Methodology for Ambient Intelligence Environments

Anna Chambers

Menai Bridge, UK
Email: absoluteconnections@hotmail.com

Received 4 February 2014; revised 1 March 2014; accepted 8 March 2014

Abstract

The current paper introduces an approach to a development of Ambient Intelligence domain-based software systems from scratch. The presented approach is based on models. The paper also presents the domain-related models expressing different levels of abstractions and stages of the development. The approach refers to a Model-Driven Development of Ambient Intelligence which was suggested at AmI-07-Ambient Intelligence conference. The approach is presented as a standard with its feasible realization. It starts from modeling of a content of the future AmI-dedicated software system and concludes by mapping the graphical concepts into a final code. A process proving feasibility and correctness of the approach is provided through a dedicated research methodology. Its process comprises an identification of needs in a speedy development of the systems. It is followed by studying of the currently available techniques capable of supporting the development and an experimenting with them. It continues by finding a solution, verified by its validation and concludes by an identification of the further perspectives. The developed approach presents a common way of a communication amongst stakeholders participating in creating of AmI-based environments. Such communication involves the notations of AmI-Creator—a Domain-Specific Language of Ambient Intelligence domain. Every part of DSL corresponds to a demonstration of A-methodology expressing a step-by-step guidance for the development. The methodology comprises two parts dedicated to providing semantics for DSL through studying of Ambient Intelligence domain ontology; and development of actual environments. A validity of the working proposition is confirmed by three examples. The further challenges refer to an extension of the presented work by other frameworks and expansion to a development of different domains with complex organizations.

Keywords

Ambient Intelligence, DSL, MDD, SE, AmI-E Development

1. Introduction

Consumers' needs in device-based sensitive environments caused a discovery of *Ambient Intelligence* (*AmI*) paradigm. Its development became a subject of a scientific discovery. The development had been discussed at AmI-dedicated events one of which suggested an involvement of Model-Driven approaches to it[1].

As a response to the suggestion this paper presents a feasible solution, which shows an easy way of building of AmI-based environments through use of *Model-Driven Development* (*MDD*)[2]. The solution demonstrates a simplification of an AmI-dedicated software development process. It starts from a consideration of a domain of interest and finalizes by mapping of modeling elements into a final code. Referring to the idea the current research articulates its question as an investigation of applicability of MDD to the AmI development process. It also tries to prove a hypothesis that such approach is feasible. The solution concerns two concepts: the third wave of computing and the fourth generation of software development. The third wave of computing refers to Ambient Intelligence whereas the fourth generation of software development refers to a modeling process.

The current paper demonstrates a development process of *AmI environments (AmI-E)* from scratch. It is based on *Model-Driven Architecture*[3] (*MDA*) initially introduced by *Object Management Group*[4] (*OMG*) and; consideration of different Software development methodologies. It demonstrates an inference of a common way of communications amongst the stakeholders participating in the software development process expressed in notations of *AmI-Creator—Domain-Specific Language* (*DSL*) for the development of AmI-based environments (AmI-E). The semantics of the language syntax corresponds to the semantics of the AmI concepts. Such concepts are presented in ontology of AmI domain expressed in various notations for their future reuse. For example they can be used for an initial analysis of the problem or for expressions used in the final code.

This paper also shows a set of models expressing different aspects and stages of the AmI-E development process. These stages are summarized in A-methodology structure of which is based *on A-MDA* (*Adapted MDA*) and MDD. A use of a MDD-based methodology allows a production of an entire flow of the software development process. It starts from an identification of the requirements and finalizes with the end code expressed in (Java) programming language notations. The main outcomes of the presented research comprise:

- Adapted Model-Driven Architecture (A-MDA).
- Methodology for the development of AmI-E from scratch.
- Ontology of AmI environments.
- AmI-Creator-DSL for the development of AmI environments.

The feasibility of the presented research is verified, validated at the appropriate stages and tested on different applications.

The current research presents a standalone idea, equivalent of which was not met in the preceding literature survey. The research evaluates the survey's finding and brings forward its argument declaring AmI as a separate domain on its own rights. This domain has the common indistinguishable parts. The development of the software-based systems from this domain can be based on a typical methodology.

The reminder of this paper shows the reason why the new approach to the AmI development is required. It also describes how it is created, realized, verified, evaluated and tested. The paper suggests a use of such approach for other domains. The paper's content covers the descriptions of:

1) Approach to the current research.
2) Overview of the existing AmI and SE related developments.
3) Demonstration of an adaptation of MDA.
4) AmI domain concepts, ontologies and DSL.
5) Description of the horizontal set of models.
6) Evaluation (through Verification, Validation and Testing).
7) Conclusion and further work.

2. Approach to the Current Research

This part describes an analysis of a central research problem and its focus. It also shows a structure of the pro-

[1]AmI-07 at http://ami-conferences.org/2007/index.html on 09.01.2011

[2]Model-Driven Development at http://msdn.microsoft.com/en-us/library/aa964145.aspx

[3]OMG Model Driven Architecture at http://www.omg.org/mda/

[4]http://www.omg.org/

posed solution and a description of a research methodological approach.

2.1. The Research Problem

The current research sees a realization of the stated idea through two contexts. The first refers to MDD. The second refers to AmI domain and approaches to its co-related developments (e.g. system development).

A preliminary consideration of MDD showed that it involved such themes as *Model-Driven Architecture* (*MDA*) and *Domain-Specific Language* (*DSL*). MDA refers to a structure, whereas DSL implies various notations of different concepts.

An initial studying of AmI domain discovered two core subjects comprising *Conceptual model* and *Ontologies*. Conceptual model comprises a set of interconnecting concepts. Ontology provides semantics and notations of the concepts and their interconnections in different representations. Ontology is dedicated to a specific reuse for specific purposes. Concretely, ontology of AmI domain comprises a set of various features characterizing the domain; which current research considers as a specific physical entity (or environment).

A further consideration of the existing AmI-related research indicated a lack of common (typical or standard) approaches to a development of AmI systems. Regarding to these, the current research declares a hypothesis asserting an existence of the approach. It shows the proof of a feasibility of such hypothesis.

The main focus of the current research concerns a development of AmI environments based on Model-Driven Development. The research puts forward an idea of using models for the development of AmI. It argues that the research proposes a standard way of intercommunications among stakeholders participating in the development of AmI software systems from scratch.

2.2. The Proposed Solution

A realization of the hypothesis peruses two separate objectives. The first refers to a consideration of Ambient Intelligence as a domain characterized by a set of common entities. Such entities refer to a set of basic elements participating in building of AmI-based software systems. The second objective refers to an obtaining of a step-by-step development of such systems.

The set of tasks leading to the objectives refers to two typical sets of activities (or methodologies). The first objective of these is to be devoted to the development of AmI domain (as a set of the standard concepts). The second is to be dedicated to the development of new AmI-based environments. The sequence of both of the sets is based on A-MDA, resulted in A-methodology comprising Vertical and Horizontal parts. The completion of the Vertical part of the methodology assures feasibility of the Horizontal part, completion of which is finalized in code.

The progression of the outlined solution requires the various activities and techniques e.g. collecting necessary information, its analysis, taking decision on the incoming tasks, verification and valuation of the data have gotten.

2.3. The Current Research Methodology

The current research methodology comprises a combination of variations of qualitative methodology, which depends on the context of the particular stage of the research. It is also updated with additional methods used by the other researches to evaluate and test the appropriate outcomes summarized in the **Table 1**.

3. Overview of the Existing AmI and SE Related Developments

This section demonstrates the latest updates in the field of AmI and general software development. The updates are summarized in three hundred twenty seven on-line publications for a period of 2005-2010. The section comprises two parts. The outcome from the first part comprises clarification of the research question. Outcome of the second part comprises providing a background for the proposed solution of the specified problem.

3.1. The First Part of the Survey Discovered the Following Issues

3.1.1. AmI-Related Overview

Ambient Intelligence is originated in a notion of calm technology or third wave of computing proposed by Mark Weiser in 1988. The purpose of calm technology is serving humans' needs with an aid of numerous computers.

Table 1. AmI research methodologies and their descriptions.

Research methodologies comprising the current research methodology	Methodologies activities
Qualitative (theoretically-based) from the published works	Collecting information, its analysis and assessment on the relevance and suitability to the research issuers
Qualitative secondary with elements of primary	To clarify the information use questionnaire and interview for the purpose of obtaining a new information (primary). Collecting the information from the presented and published works concerning different proposed solutions and their income in the studied subject (secondary)
Evaluation and verification	Use different methods of assessing the obtained results (e.g. as factors and principles). Use different interview and questionnaire techniques for the verification of the obtained results and getting feedback (possibly) showing new directions in the research

It found its implementation in such areas as health [1] [2], sport [3], administration [4], business and education [5]. Its increasing demand is indicated as a desirable involvement in applications [6] supporting care services based on modern technologies. The examples of such applications refer to such projects as "intelligent home", "personal health" and "ambient assisted living".

The AmI definitions vary. They include such notions as Ubiquitous/Pervasive/Proactive/Ambient Computing, Calm Technology and Smart Environments [7]. The interpretations and visions refer to "multidisciplinary paradigm" [8] and "vision of future Information Society" [9].

Summarizing the information ([10] [11], MIT cited in [12]-[14]), the current research indicates that the idea behind AmI refers to the set of interconnecting devices working together in the intelligent way. AmI can serve humans' tasks of "putting a technology on people's services and for their benefits" [13]. The devices facilitate an electronic (or digital) environment. The information indicates a sharing of the basic assumptions about AmI properties and a way of their composition for the behaviors in an intelligent or "smart" way. For example the properties of ubiquity and disappearance refer to challenges of design interactions in future smart environments [7].

The ubiquity refers to an increasing number of computers around us, their incorporation in different parts of our environments and becoming a part of our life by maintaining everyday routines.

The disappearance is explained as tendency of computing devices to get smaller and disappear (hiding) in a textile surroundings us. The incorporation of such devices refers to integration of information, communication and sensing technology into everyday life leading to turning of our environments into "smart". The "smart" concept refers to capabilities of sensing, processing, making decision and actuation, which has to have self-directions (e.g. actions or re-action).

The AmI has been understood as a set of entities with different arrangements such as Service-Oriented Architecture [15], structured context information AmbiNet [16], Agent-based architecture ([17] [18]) managing different technologies e.g. Wi-Fi, FRID, ZigBe or a coordinated structure of "domotic" components [19] and an architecture for an activity recognition using Visual Sensor Network [20]. The execution of AmI is supported by such technologies as a networked control system (NCS) [21], Near Field Communication Technology [22], and other [3] [23]-[26].

Some of the AmI-related themes refer to ontology of a single activity as a set of tasks [27] or ontologies describing "activities sphere" [19]. Another theme refers to a conceptual model [28] comprising layers of sensors and actuators, a network and a middleware, a device, a service and an AmI application.

Alternatively AmI was realized as middleware comprising different services and facilitating interactions of different parts [15] [29]-[31]. Further AmI-based interface is presented as adaptive interface [32] for complex AmI environments sharing resources (hardware, networks and knowledge); or providing a generation of user interfaces for Ambient Assisted Living services based on an interaction framework [33]. "World around us is interface to information" refers to AmI context [7]. An AmI context-awareness specifies a nature of a context as spatiotemporal, environmental, personal and social [34]. Ambient Intelligence have been developed by some companies (Opt.Cit. [10] [35]).

The bases for building of AmI were introduced through such examples as AMENITIES methodology [36], 3D-based rapid prototyping [37]; a Product Line Approach [38], Aspect-oriented programming [39].

3.1.2. AmI Environments Issues

To benefit the consumer, a substance of Ambient Intelligence should be attached to a physical entity or an environment. Overview of the existing AmI examples showed that previously the notions of AmI environments were used for testing. Such, a use of the environment was presented in "the rapid prototyping and analysis of Ambient Intelligence systems" [40]. Implicitly AmI environments were introduced as space of "activity sphere" [41] or "AmI Ecosystem" [42]. Further, there were no indications of AmI as a domain on its own rights. There also were no indications of a standard approach to the development of AmI-based software systems.

***As things stand above*:** Previously Ambient Intelligence was presented as a vague technology-based essence. It was understood as comprising element or a summation of elements or disciplines. Regarding to the remaining text, the current research reckons that use of term "Environment" is more appropriate to an identification of "Ambient Intelligence" concept as referring to a particular space under an influence of AmI. Ambient Intelligence represents an "Intelligent environment which reacts in a sensitive and adaptive way to the presence of different objects in order to provide various services to people" [43].

Having comprehended the information about AmI the current research provides its own definition of Ambient Intelligence as:

Ambient intelligence *is Artificial Intelligent Technology-based Environment providing responses on the dynamically detected requests and satisfying the predefined tasks in the expected way and in the expected time.*

Furthermore, Ambient Intelligence was articulated by AmI-08[5] as a subject of expansion into wider areas. Additionally it was indicated that "engineering such systems is a nontrivial task, and convincing solutions are still missing" [44]. In response to such statement the researches provide different ideas, examples of which comprise an idea of attracting SE (MDD in particular) into the development of AmI. Such idea was articulated at AmI-07. Another idea of using Software Engineering for the development of Ambient Intelligence was suggested at AmI.d-07[6].

Finally the research question is clarified as focusing on proposition of the standard MDD to the development of AmI systems.

3.2. The Second Part of the Survey Discovered the Issues Clarifying What Is Exactly Involved into the Software Engineering Notion of Model-Driven Development of AmI-E

The current research identifies context of models as a clear set of formal elements that describe something being developed for a specific purpose and can be analyzed using various methods [45]. Engineering model must include the characteristics of Abstraction, Understandability, Accuracy, Predictiveness and Inexpensiveness.

3.2.1. MDD and SE Concerns

MDD is associated with a notion of Model-Driven Architecture (MDA), which provides a basic terminology for MDSD—Model-Driven Software Development. The current research sees MDD as means supporting a collaborative discussion among the project's stakeholders for obtaining an immediate feedback.

There is no agreement among the SE researches on a precise definition and content of MDA. Some of them see MDA as a new approach to writing specifications and application's development. The developments are based on a Platform-Independent Model (PIM)[7] of the application or specification of business functionality and behavior. The other [46] explains MDA as "about using modeling languages as programming languages rather than merely as design languages". OMG clarifies principles of MDA as (1) the understanding of systems for enterprise solution laying in the models with well-defined notations; (2) the architectural framework of models, comprising layers and transformations structure, can serve the purpose of building systems; (3) the tools-based basis for an automotive models integration and their transformation; (4) this Model-Driven approach is to be based on industry standards, openness to consumers and encouraging vendors competitions.

The original MDA OMG is organized into a four-layer architecture [45] which comprises:

- M3 layer meta-meta-model: MOF-Meta-Objects Facility.
- M2 layer meta-modeling: UML profile Custom meta-model.
- M1 layer model: UML models, Custom meta-model, Models based on the custom's meta-models.
- M0 layer reality: The Real World.

[5]AmI-08 Conference in Nuremberg at http://www.ami-conferences.org/2008/index.html
[6]AmI.d 2007 at http://www.strategiestm.com/conferences/amid/07/index.htm
[7]http://www.theenterprisearchitect.eu/archive/2008/01/16/mda-model-driven-architecture-basic-concepts

Object Management Group points out on a modeling provided with OMG's standards. One of these is Unified Modeling Language (UML) which is stated as modeling language of choice [47], and also points on a tendency of using MDA framework for a Domain-Specific Modeling defining Domain-Specific Languages.

"Domain-Specific Modeling (DSM) is a model-based software development approach that uses visual models as primary artifacts in the development process" [48]. DSLs should have notations, which are understandable by the people involved into a development. "Modeling provides abstractions for specifying problems and solutions in more productive and effective manner" [49]. OMG points on MDA as next step in the modeling evolution of MDD technologies. To implement the idea of MDD as a set of working representations requires an *Integrated Development Environment* (*IDE*).

Research Note

The current research sees involvement of MDD as a framework supporting a step-by-step development based on a purposely developed methodology. Referring to the idea of using MDA for the entire application's development the current research sees the MDA as a base for summarizing the entire requirements for the development of AmI applications. Such requirements comprise modeling notations and software tools supporting the models' production.

3.2.2. Software Methodologies

Further, SE offers two types of methodologies referring to a step-by-step development process. It also includes such approaches to a general software development, a development of AmI software systems and DSL. The SE methodologies refer to Waterfall, Evolutionary development, Rational-unified process, V or Agile methodologies.

3.2.3. AmI-Dedicated Methodologies

The examples of AmI dedicated approaches and methodologies refer to "Design for all" [50], Universal Access perspective [51], Software product line approach for AmI environments [52]. Other examples ([14] and [53]) outline a process of building AmI environments: starting from needs of sensors and devices for surrounding the environmental inhabitants with technology, where sensors collect different context information which is to be transmitted by a network and pre-processed by middleware. The examples do not describe a sequence of the activities for building AmI environments. Other examples comprise cooperative system design [54] and design framework [13].

The example refers to rapid prototyping of creating an AmI demonstrator [29]. User-Centered design [55] put forward message: "The design interactive systems, that are targeted to be introduced into the market within a timeframe of five to ten years, remains a methodological challenge". The current research considers this as a request to a new methodology.

Despite the variety of the approaches, AmI scientists ([56] [57]) state that clear methodology for digital ecosystem (or AmI) design does not exists yet. It gives an interesting hint on De Ruyter's approach through building prototypes to be discussed with stakeholders.

3.2.4. DSL Methodology

The methodologies proposed for the development of DSLs comprise several activities [58], [59]. The current research agrees that prior to the language development the domain should be studied, but it does not agree that the development should start from the syntax definition. In the current research opinion, the developer should study domain initially. It should be followed by an extraction of the semantics corresponding to the appropriate syntax, expressed in different concrete notations. For the current research purpose the notion of DSL is preferable to general purpose programming languages as it can provide domain-specific notations to be used for modeling of the different stages of the development. The content of the stages should be discussed with customers and stakeholders.

3.2.5. Request to New Methodologies and AmI Development

Despite approaches and methodologies for AmI-related development are already indicated by some of the authors, the other provide request to a methodology for AmI-related developments. The last indicates a lack of models and software engineering practice helping in managing of a AmI development lifecycle [34]. It points out on needs of methodologies for development; on possible involvement of direct software engineering methods suitable for the development of request-response systems with strictly defined behavior. Further the authors indicate that: "The request-response scenario is replaced here by continuous sensitive, reactive, intelligence surrounding computing." The current research disagrees with such statement and argues that nature of Ambient Intelligence is in providing the appropriate response on the identified request (which is obtained through detections of the appropriate environmental conditions).

3.2.6. Tools and Languages

Some researches show tools [60] used for developments in AmI such as Ambi Graphs. The syntaxes comprise such notations as boxes, rhombus and arrows. Authors indicated a necessity of the context modeling, analysis and reasoning (cited in [61]). Models are good for expressing requirements, reasoning and discovering anomalies, inconsistency and incompleteness.

As things stand above: The proposed solution of MDD in AmI can be realized by means of MDA, explaining the way of obtaining semantics and syntax for DSL called AmI-Creator. It can be supported by the appropriate tools; and methodology comprising the set of steps necessary for the development of AmI systems.

4. Demonstration of Adaptation of MDA

Believing that the development of AmI-E can be unified, standardized and presented as modeling process based on Software Engineering notions of MDA and MDD the current research generates the research hypothesis that:

"*The Model Driven Development approach to the development of AmI domain and applications, based on MDA is possible and can be produced*".

Such hypothesis refers to an inference of a way dedicated to a development of AmI based system from scratch. Focusing on finding a common way of the communication among the stakeholders during the AmI development process the current research puts forward an idea of models. Their semantics should be identified by ontologies. Their syntaxes should be expressed in DSL notations.

The inference of DSL and its concrete use the current research sees through an adaptation of an idea behind MDA. It was generated by the organizations interpreting MDA as an approach to application design and implementation. "As defined by the Object Management Group (OMG), MDA is a way to organize and manage enterprise architectures supported by automated tools and services for both defining the models and facilitating transformations between different model types."[8] In respect to the generated idea of MDA-based structural approach the author presents a novelty to a creation of AmI applications from scratch. It refers to a realization of the requirements comprising models and tools supporting their production. This idea is developed by the current research.

The hypothesis corresponds to the appropriate theory that MDD is a right approach to the development of AmI-based systems. The current research sees the proof of the theory through a conjunctive consideration of two research fields comprising SE and AmI. The first of them serves the development of the second.

Supporting its theory the current research concentrates on producing the following outcomes:

- Outline, structure, production and implementation of the Model-Driven Development Methodology for creating software applications from AmI domain from scratch
- Present AmI-Creator—a new Ambient Intelligence-dedicated Domain Specific Language (AmI-d-DSL)
- Realize IDE (Integrated Development Environment) for AmI, implementing the DSL for AmI

The intended results of the development imply to bring the benefit to the AmI development stakeholders, whether they are developers or not. This benefit would provide the user with a fully developed flow of the development of AmI applications (finalized in the code realization), supplied with AmI-dedicated language and realized tools.

The current research sees a production of the desired targets as outcomes of a "Walkthrough" of the Adapted MDA (A-MDA) based on the structuring idea behind MDA OMG. From the original idea of MDA the current research adapted only the layered arrangement of solving a problem. The every level of A-MDA is dedicated to a task e.g. domain analysis or study the requirements for the appropriate tools. Comparison of two architectures is shown in **Table 2**. Despite the similarities of the layers the MDA and A-MDA are differentiated. A-MDA does not pursues the purpose of MDA producing a model of the immediate lower level; but pursues a single purpose of comprehension and sufficiency of needs for modeling and building of AmI environments:

[8]http://www.ibm.com/developerworks/rational/library/3100.html

Table 2. Structure of A-MDA (Adapted MDA OMG).

Level	MDA OMG	A-MDA	comparison
M0	Real world	Real world	Same
M1	Models of the World	Models of the World	Same
M2	Meta-model	Meta-model	Same
M3	Meta-meta-model	Tools definition	Different

1) The lowest level is devoted to a studying of the dedicated domain.
2) The next level up derives modeling elements of the real world.
3) The following level derives basic modeling concepts.
4) The top level realizes the tools' requirements.

The A-MDA provides a structure for definition of two methodologies comprising methodology for creating of AmI-based DSL (called AmI-Creator) and methodology for the development of software systems from AmI domain.

In respect to the development of AmI-Creator the requirements for A-MDA refer to a providing a base for outlining the process of the development of DSL for AmI environments. It comprises an obtaining of the DSL semantics, a concrete syntax and its supporting tools. The process corresponds to a methodology outlining the sequence of activities. They start from analysis of a domain with extracting a vocabulary (refer to ontology), continue through design (modeling) and finalizes by realization of the supporting tools. The process is based on a technique called "Walkthrough" referring to the step-by-step activities provided at A-MDA levels:

- In the reference to M0 (real world of AmI environment):
    - Studying the world of Ambient Intelligence Environment.
    - Derive the comprising concepts, their relations and intercommunications.
    - Summarize findings of studying the domain in (middle) ontologies.
    - Extract domain vocabulary (or semantics).
    - Document the findings as ontology documents.
- In the reference to M1 (model of AmI environment):
    - Outline the sequence of models and their contents.
    - Extract the appropriate semantics expressed in vocabulary for every specified model.
    - Specify concrete syntax.
    - Document the findings.
- In the reference to M2 (meta-model of model of AmI Environment):
    - Analyze domain of models of the level M1 and derive the set of models' comprising concepts.
    - Summarize, generalize and abstract them into the general types of modeling entities, capable of creating of any modeling concepts for models in the M1.
    - Identify these entities as general types, comprising the concepts (or upper-ontology).
    - Define the upper ontology as vocabulary for the next part of AmI-Creator (AmI-DSL).
- In the reference to M3 (as realization of tools to depict syntax):
    - Analyze requirements ensuring full support of the modeling features.
    - Give detailed requirements of the working tools.

There are two kind of "Walkthrough" were considered at the research: Vertical and Horizontal. Vertical "Walk through" (**Figure 1**) contains two parts: bottom-up (inferring requirements) and top-dawn (for realization and implementation of the requirements). Horizontal "Walkthrough" (**Figure 2**) has left-to-right direction outlining the activities for an application development (at M1) and mapping of the requirements into their realization (at M0, M2, M3).

The Methodology Description

The realization of the current research argument comprises two perspectives. They refer to an obtaining mea-

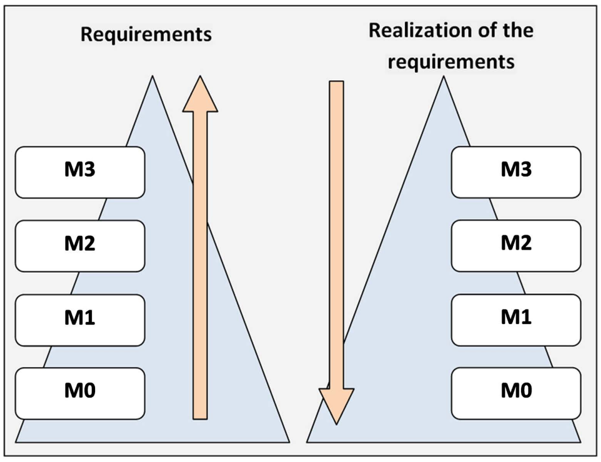


Figure1. Vertical "Walkthrough".

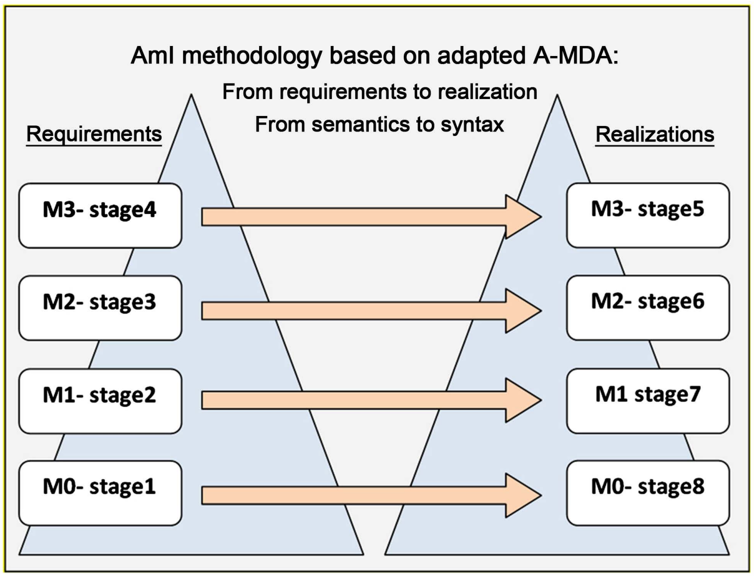
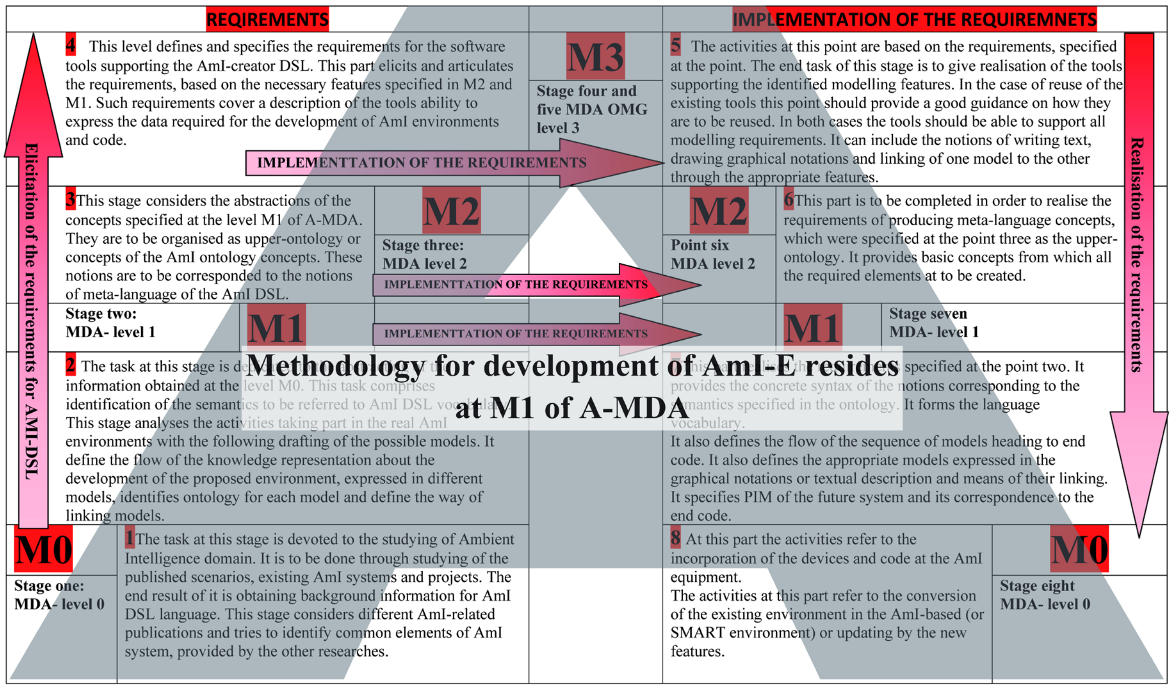


Figure 2. Horizontal "Walkthrough".

ningful notations and a consecutive development of activities; where the first refers to a DSL and second refers to a methodology.

The inference of the notations complies a vertical "Walkthrough" of A-MDA. It occupies three bottom layers (M0 - M2) and comprises activities of bottom-up (from M0 to M2) and top-dawn (from M2 to M0) directions. Bottom-up activities are dedicated to obtaining of AmI domain concepts, suggestion of language semantics, their abstractions into meta-language semantics and finalized with a deriving of tools requirements. The bottom-up part also suggests the graphical notations. The top-dawn activities comprise: the realization of the appropriate tools capable of holding the appropriate graphical notations, choosing notations for meta-language and DSL. The horizontal part of A- methodology resides at M1 layer of A-MDA and derives the stages for the development of new AmI domain-based applications. The overall logical structure of the AmI-dedicated development from scratch forms a shape of capital "A" (Figure 3), after which the methodology was called. The Vertical bottom-up and top-dawn directions provide a background for the implementation of the Horizontal part.

The summary explains that the Vertical part is dedicated to the development of AmI domain and its modeling activities, whereas Horizontal part is dedicated to the development of AmI software systems. As overall the A-methodology is dedicated to the development of AmI software application from scratch. It means a starting from studying a domain and realization of the supporting tools for creating applications.


REQIREMENTS
IMPLEMENTATION OF THE REQUIREMNETS
Elicitation of the requirements for AMI-DSL
Realisation of the requirements
M3
Stage four and five MDA OMG level 3
IMPLEMENTTATION OF THE REQUIREMENTS
M2
Stage three: MDA level 2
M2
Point six MDA level 2
IMPLEMENTTATION OF THE REQUIREMENTS
M1
Stage two: MDA- level 1
IMPLEMENTTATION OF THE REQUIREMENTS
M1
Stage seven MDA- level 1
Methodology for development of AmI-E resides at M1 of A-MDA
M0
Stage one: MDA- level 0
M0
Stage eight MDA- level 0

4 This level defines and specifies the requirements for the software tools supporting the AmI-creator DSL. This part elicits and articulates the requirements, based on the necessary features specified in M2 and M1. Such requirements cover a description of the tools ability to express the data required for the development of AmI environments and code.

3 This stage considers the abstractions of the concepts specified at the level M1 of A-MDA. They are to be organised as upper-ontology or concepts of the AmI ontology concepts. These notions are to be corresponded to the notions of meta-language of the AmI DSL.

2 The task at this stage is d[illegible] of the information obtained at the level M0. This task comprises identification of the semantics to be referred to AmI DSL vocabulary. This stage analyses the activities taking part in the real AmI environments with the following drafting of the possible models. It define the flow of the knowledge representation about the development of the proposed environment, expressed in different models, identifies ontology for each model and define the way of linking models.

1 The task at this stage is devoted to the studying of Ambient Intelligence domain. It is to be done through studying of the published scenarios, existing AmI systems and projects. The end result of it is obtaining background information for AmI DSL language. This stage considers different AmI-related publications and tries to identify common elements of AmI system, provided by the other researches.

5 The activities at this point are based on the requirements, specified at the point. The end task of this stage is to give realisation of the tools supporting the identified modelling features. In the case of reuse of the existing tools this point should provide a good guidance on how they are to be reused. In both cases the tools should be able to support all modelling requirements. It can include the notions of writing text, drawing graphical notations and linking of one model to the other through the appropriate features.

6 This part is to be completed in order to realise the requirements of producing meta-language concepts, which were specified at the point three as the upper-ontology. It provides basic concepts from which all the required elements at to be created.

7 This part realises the requirements specified at the point two. It provides the concrete syntax of the notions corresponding to the semantics specified in the ontology. It forms the language vocabulary.
It also defines the flow of the sequence of models heading to end code. It also defines the appropriate models expressed in the graphical notations or textual description and means of their linking. It specifies PIM of the future system and its correspondence to the end code.

8 At this part the activities refer to the incorporation of the devices and code at the AmI equipment.
The activities at this part refer to the conversion of the existing environment in the AmI-based (or SMART environment) or updating by the new features.

Figure 3. Depiction of inference of A-Methodology.

5. AmI Domain Concepts, Ontologies and DSL

5.1. AmI Domain Concepts

This part demonstrates the step-by-step analysis process. It comprises three stages of initial analysis of the studied domain, ontologies as intermediate analysis of the domain for it future reuse. It finalizes by using the ontology's concepts as semantics corresponding to the concrete syntax of the concrete DSL. It shows an inference of the modeling concepts to be used in the actual modeling of AmI environments.

Having studied the AmI examples the current research realizes AmI domain as an entity to be considered from two perspectives. The first perspective refers to the request-response context (**Figure 4**). The second perspective refers to a conceptual model. The last comprises main sets of concepts abstracted into two groups of time-spatial and software components (**Figure 5**). The request-response perspective implies that some entity perceives a data (or context of the environment). This is specified as a request. The other entity interprets the context and generates an appropriate response. The context model perspective implies an obtaining of a set of the domain concepts (see left part of **Figure 6**). These are considered as a base for the corresponding AmI ontologies (see middle part of **Figure 6**). This perspective demonstrates the way of studying of the domain semantics (in a conceptual model and ontologies). They are followed by their expressions in the appropriate language and meta-language referring to the appropriate syntax (see right part of **Figure 6**).

Having considered and compared the notions of Conceptual model, Ontologies and Domain-Specific Language the current research found that all three notions showed that they are very close in their ideas. All of them collect and express the domain information, but pursue different purposes:

- Concept model is used for studying a domain.
- Ontology express a domain concepts and their interconnections in different notations.
- DSL provide notations for creating of different applications.

These findings are summarized in the **Figure 7**, which shows the boundaries in dashed lines. The figure shows that Conceptual model of AmI domain comprises Static and Dynamic parts. The Static part refers to the concept of an Object and its Characteristics; whereas Dynamic part refers to a concept of Process. The concepts (comprising Object, Subject and Process) which are identified in the Conceptual model refer to the concepts with the same names of the Upper ontology of AmI domain. They also refer to the meta-modeling concepts of a future Domain Specific Language.

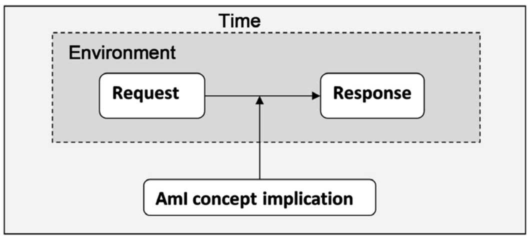


Figure 4. AmI request-response context.

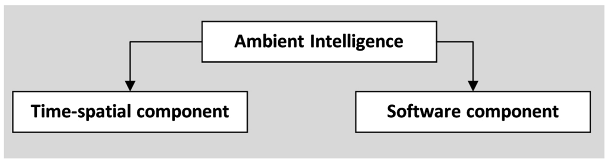


Figure 5. AmI second concept.

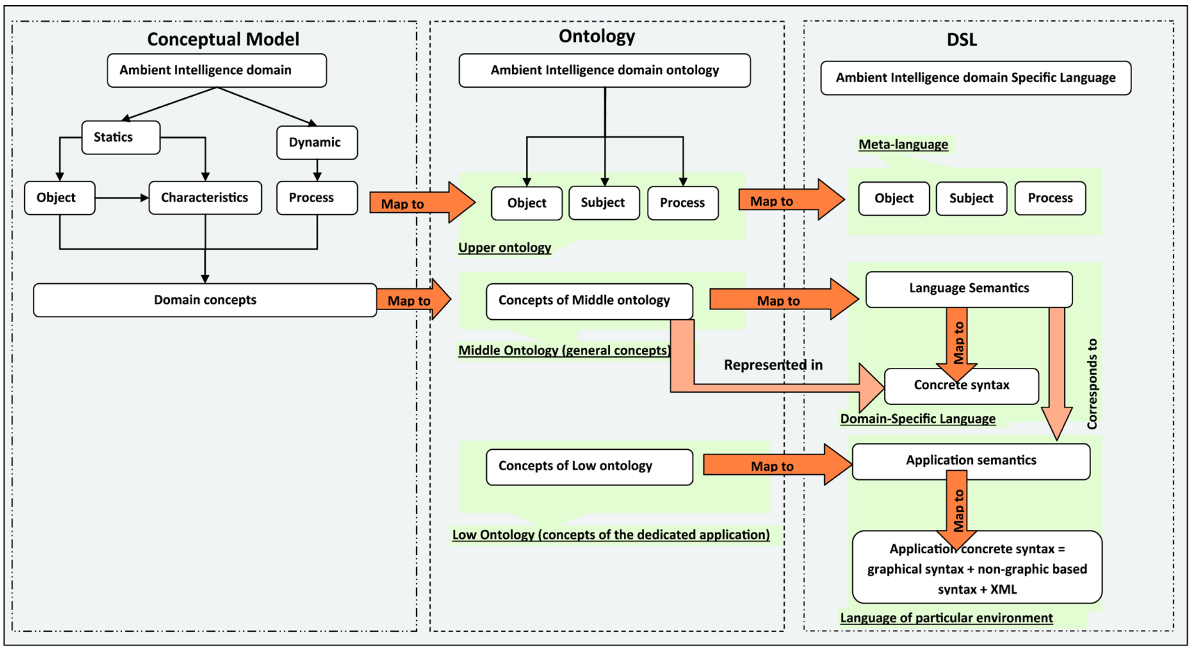

Conceptual Model
Ambient Intelligence domain
Statics
Dynamic
Object
Characteristics
Process
Domain concepts
Map to
Ontology
Ambient Intelligence domain ontology
Object
Subject
Process
Upper ontology
Concepts of Middle ontology
Middle Ontology (general concepts)
Represented in
Concepts of Low ontology
Low Ontology (concepts of the dedicated application)
DSL
Ambient Intelligence domain Specific Language
Meta-language
Object
Subject
Process
Language Semantics
Map to
Concrete syntax
Corresponds to
Domain-Specific Language
Application semantics
Application concrete syntax = graphical syntax + non-graphic based syntax + XML
Language of particular environment

Figure 6. Interconnections between conceptual model, Ontology and DSL.

Three concepts of the Conceptual model represent the abstractions of the AmI domain concepts, derived during an initial analysis of the domain. They correspond to the concepts of the Middle Ontology. This ontology is mapped into a DSL's semantics and are represented in the concrete syntax of DSL.

The concepts of the Middle ontology refer to the abstractions of the Lower ontologies, which are mapped into the application's semantics of a future concrete system. The application semantics correspond and map into the concrete syntax of the Domain-Specific Language which can be represented in graphical and non-graphical notations.

One of the non-graphical notations refers to a XML notations, which provide a background for model transformation (refer to the further research).

5.2. Ontology

The purpose of AmI ontologies refers to their further reuse. Depending on the reuse the ontologies can be expressed in more than one notation and can be represented at different levels: Upper, Middle, Lower, Lowest

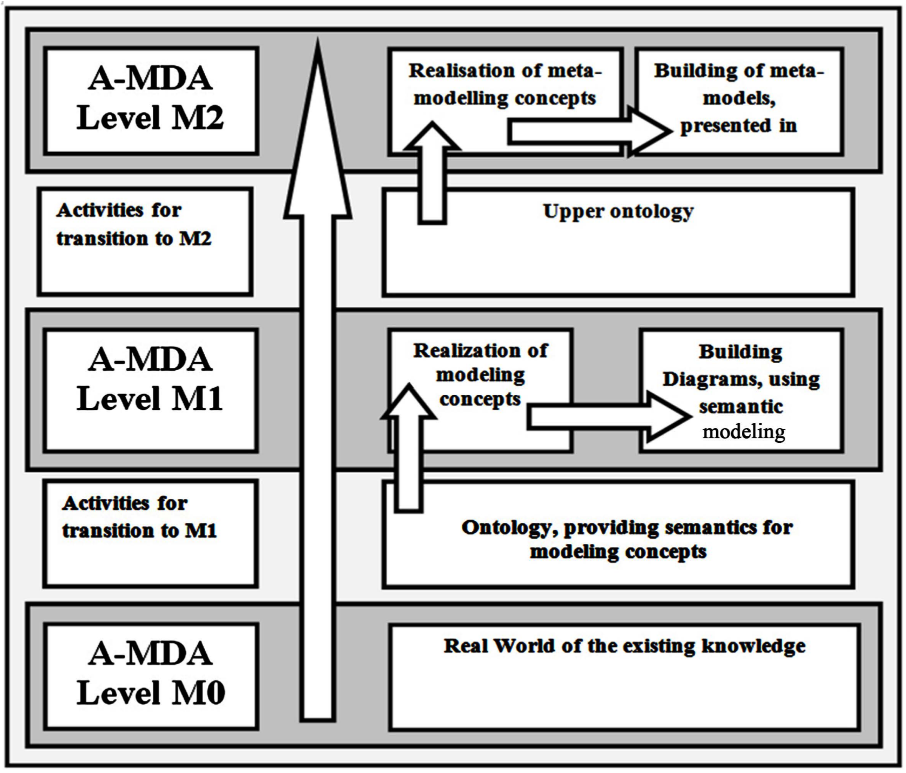

A-MDA Level M2
Realisation of meta-modelling concepts
Building of meta-models, presented in
Activities for transition to M2
Upper ontology
A-MDA Level M1
Realization of modeling concepts
Building Diagrams, using semantic modeling
Activities for transition to M1
Ontology, providing semantics for modeling concepts
A-MDA Level M0
Real World of the existing knowledge

Figure 7. Description of entire activities at A-MDA including obtaining ontologies.

[62]. The current research interest concerns Upper and Middle ontology representing the AmI domain modeling concepts, where Upper ontology considers Generic Common Knowledge and comprises most General Things; Middle Ontology considers Domain Spanning Knowledge.

The way of inference of the ontologies is depicted in the (Figure 7). It demonstrates a use of A-MDA in respect of showing the activities at the appropriate levels of the A-MDA. Such, it shows that obtaining the ontology (refer to middle ontology) of the real world of AmI domain is provided through a studying of the existing AmI representations and applications mentioned in Section 3.

The studying extracts the meaning of the concepts, which were met in the existing examples. Such meanings provide the semantics for DSL-AmI-Creator. Obtaining the Upper ontology is provided through an abstraction of the middle ontology. Its purpose concerns a providing the semantics for basic modeling concepts, which refer to the meta-models and meta-language concepts. Apart from these the A-MDA ensures the activities for obtaining the requirements for the software tools. The tools should support the feasibility of the producing the notations. The overall approach to an extracting of the semantics for the ontologies from the existing examples is based on the motto:

"*Summarize the commonalities and distinguish the differences of AmI-based entities*".

All of the examples showed an existence of the same concepts, containing such notions as Environment (in capacity of a holder of AmI content), Devices intercommunicating through Network, Context providing different information about the environment and Time as length of some kind of a process or performance. These refer to the properties of AmI domain and present a set of the standard characteristics of AmI domain.

The nature of AmI environments is characterized by two contexts of the physical (or time-spatial) entities and software. The successful work of the AmI environments is supported by the appropriate software. Referring to it the ontology comprises software and time-spatial components (Figure 6). The ontology of the AmI-E software context is represented at the picture Figure 8. It shows that Software context comprises:

- Software residing on the Devices including:
    - Sensors' software.
    - Actuators' software.
    - Sensor-actuators' software.

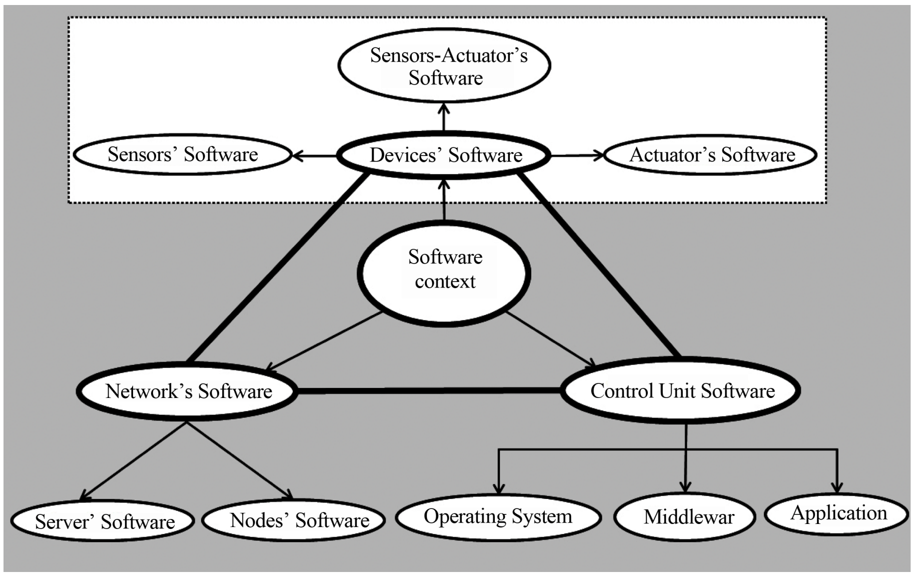


Figure 8. Ontology of the AmI-E software context.

- Software supporting a working Network ensuring a successful work of the interconnecting devices including:
  - Servers' software.
  - Nodes' software.
- Software supporting a work of a Control Unit of the entire AmI system including:
  - Software of an Operating system.
  - Software of a middleware supporting the logic of the system.
  - Software of probable applications e.g. modulator/demodulator.

The software component does not participate in the actual building of AmI environments will not be considered in details in this paper. The ontology of the time-spatial context refers to the main focus of building of AmI environments. It comprises a set of the components interconnecting through interfaces as shown in **Figure 9**. It also depicts the appropriate intercommunications through interfaces. It shows that the concepts of this context comprises:

- Environment.
- Network.
- Devices.
- Time.
- Context of the Environments referring to its different properties.

These concepts (depicted as ellipses) communicate with each other through interfaces, which can be of hands-free (depicted in dashed lines) or direct (physical) contact (depicted in solid lines). The directions of the intercommunications are shown as the arrowed lines. For example, it is shown that Environment contains Devices, which are capable of detecting a Context. The pointing down arrow shows that the Context's data is to be processed, but the actual process is not a concern of the current paper.

The central entity of the **Figure 9** is Environment as physically containing Devices and Network and having particular properties, referring to Context. The Context is supplied the Devices with the detectable data, which is sensed and processed during a period of time. The detected data is sent by the Devices through a Network; which provides the interconnections among the Devices. The complete ontology of Ambient Intelligence environment is shown in **Figure 10**. The current research defines the structure of AmI ontologies as a nested arrangement. It implies that relations between the concepts are organized as nesting of one concept into the other (**Figure 11**). The structure is explained as the paradigm of Ambient Intelligence having a limited influence and limited to the certain boundaries. The substance(s) inside of the boundaries refer to AmIHolder, which contains at least one environment. The environment is characterized by different features (Context) which can be detected by the dedicated devices. The devices are capable of intercommunication through a network. The active-

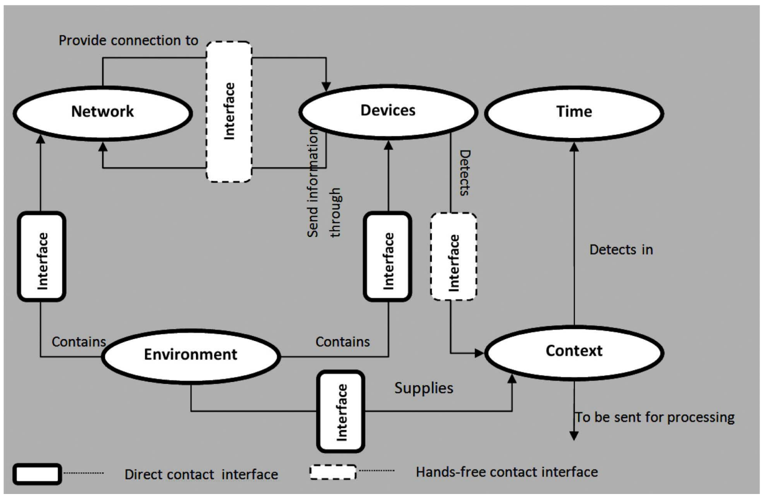


Figure 9. Ontology of the AmI-E Time-spatial context.

ties are taking place during some period of time and refer to the notion of process or performance. The research interpretation of the AmI domain is presented in the model of middle ontologies and upper ontologies. They are expressed in forms explaining that Middle ontology of AmI comprises Holder/Environment, Device, Context, Network, Time and Process. The middle ontology is referred to the expression:

$$\textbf{AmI} = \textbf{Holder}/\textbf{Environment} + \textbf{Device} + \textbf{Context} + \textbf{Network} + \textbf{Time} + \textbf{Process}$$

The upper ontology of AmI comprises elements of Object, Subject and Process:

$$\textbf{AmI} = \textbf{Object} + \textbf{Subject} + \textbf{Process}$$

The every concept comprising the ontologies has its own properties and can be considered as a domains on their own rights, example of which is depicted in the **Figure 12**. This particular ontology is intent to be used for analysis purpose. For use in programming the some ontology was expressed in Java notations for example:

```
public class Environment{
private String Name;
private int ID;
private String type;
AmIMeasurements [][][] amMeasure;
AmICoordinates [][][] amICoord;
AmIContext[] amIContext;
AmIDevice[] amIDevice;
AmINetwork[] AmINetwork;
AmITime[][][] amITime;
pulic r_type performance()
{//methods can referto UserRequest}
```

Regarding to the both of the contexts the research presents a mathematical representation of AmI domain (**Figure 13**). The picture explains that response is generated as reaction provided by the system on a particular state of the environment. The state is based on the particular values of the environmental characteristics referred to the context. The equitation demonstrates a general representation of the idea behind a request-response concept of AmI.

It is expressed as a adequateness of the result f(a) at the right-hand side of the equitation to the requests obtained as a sensed data by a set of sensors. It tells that every response (or action) f(a) is provided on the identified request, obtained as output from the set of sensors (s) number of which varies from none to indefinite. Further the current research states that every action or a particular event (a) is taking part in a response to a particular request (s) in the particular environment (e) and at the particular time (t) as shown in the Statement 1: $\forall a \exists s,e,t$. The current research argues that it is possible to apply the mathematical model to an expression of the entire AmIHolder. The research speculation states that Ambient Intelligence can be executed in some kind of

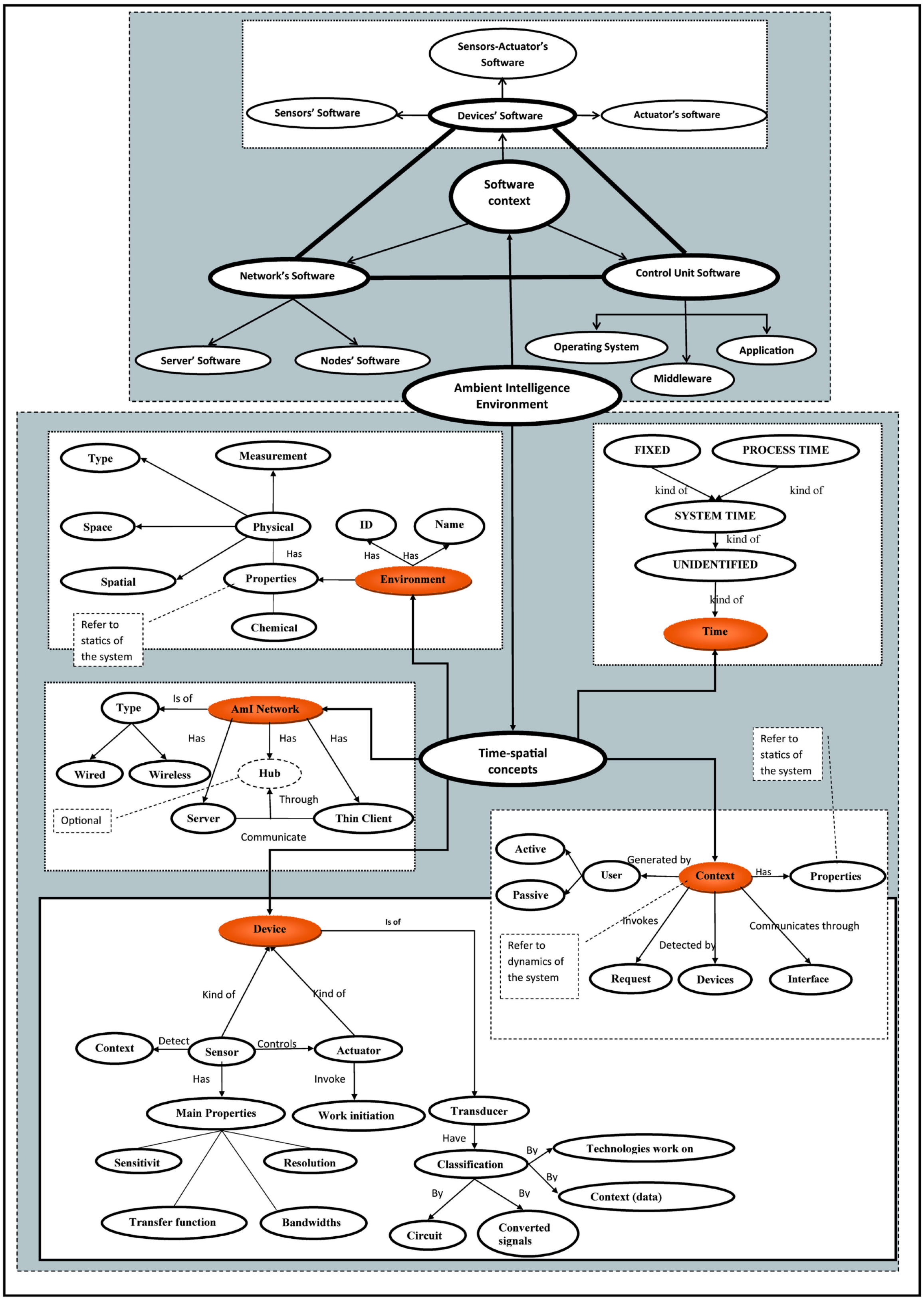


Figure 10. The complete ontology of Ambient Intelligence Environment (depicted in the middle) comprising Software component (depicted as Software context) and Time-spatial component (depicted as Time-spatial context).

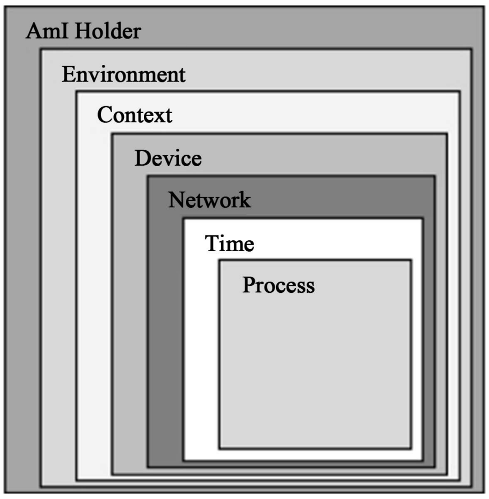


Figure 11. Nested ontology of AmI-E.

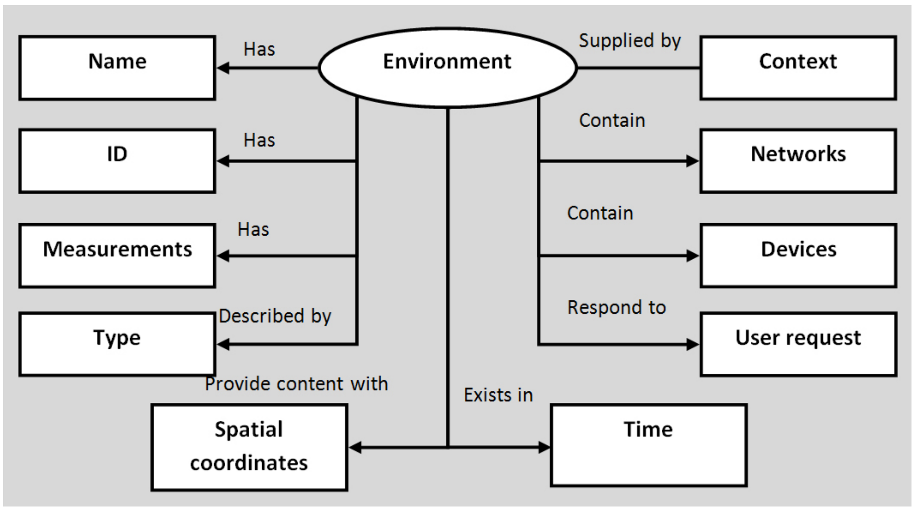


Figure 12. Ontology of environment concept.

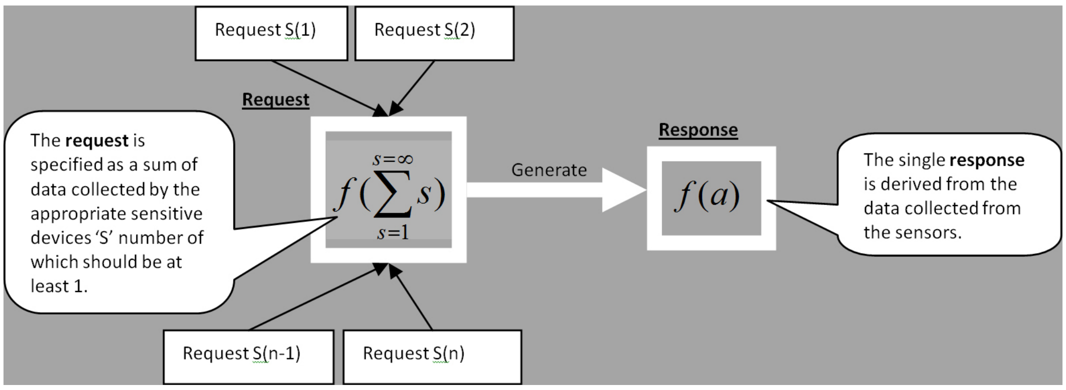

Request S(1)
Request S(2)
Request
The request is specified as a sum of data collected by the appropriate sensitive devices 'S' number of which should be at least 1.
$f(\sum_{s=1}^{s=\infty} s)$
Generate
Response
$f(a)$
The single response is derived from the data collected from the sensors.
Request S(n-1)
Request S(n)

Figure 13. Mathematical model of AmI domain.

physical existence characterised by certain dimensions. Every sensor placed in such existence gets the data from a particular position, identified by coordinates (x,y,z) and in a particular time, which is depicted in the Equitation (1).

$$f(pi) = \sum_{e=1}^{e=\infty} e\left(\sum_{n=1}^{n=\infty} n\left(t_i, \sum_{a=1}^{a=\infty} a\left(\sum_{s=1}^{s=\infty} s(x, y, z, t_0)\right)\right)\right) \tag{1}$$

The equation should be read as follow: "The overall performance pi at the locale l comprises the performances, presented in the summarized environments e which is(are) equipped with the network(s)n, assisting the summarized number of performed actions in responses to the set of calculated requests (value of sensing data). The requestes are obtained from the sensors, allocated to the specified coordinates at the detected time t_0". This equitation indicates that AmIHolder can comprise an unlimited number of sensors, actions, networks and environments. It also refers to one of the version of the AmI ontologies represented in the form of mathematical notations. Such can be reused for the identification of the precise components of the future AmI systems, explaining the nature of AmI in the most abstract way.

5.3. DSL Concrete Syntax

The actual DSL with its appropriate concrete syntax corresponding to the development stages (**Figure 14**) of AmI-Creator (**Figure 15**) is presented in the following **Figures 16-19**. The symbolical representations of the AmI-Creator should be used in the development of AmI-E. They include graphical notations, which are employed for the analysis and constructing of the environments. The Java programming notations are used for the implementation or an actual coding. The DSL is explained in the Section 7. The actual figures refer to:

- **Figure 16** refers to: Concrete syntax of the AmI-Creator DSL for depicing of both: the environments at AmIHolder and overall performance at AmIHolder.
- **Figure 17** refers to: Concrete syntax of AmI-Creator for coarse organisation of the environments (the first, second and last row show the examples only).
- **Figure 18** refers to: Concrete syntax of AmI-Creator for realisation of the performances at the different environments.
- **Figure 19** refers to: Concrete syntax of the AmI-Creator DSL for depicing of more detailed performance at environments.

6. Description of the Horizontal Set of Models

Having provided the background for the development of AmI-E software systems, the A-methodology describes the development process at the Horizontal part of A-methodology (or Horizontal methodology). It realizes the development of AmI environments (or systems). The structural flow of such realisation resides on the level M1 of A-MDA and describes the step-by-step methodology of the complete AmI-E development process.

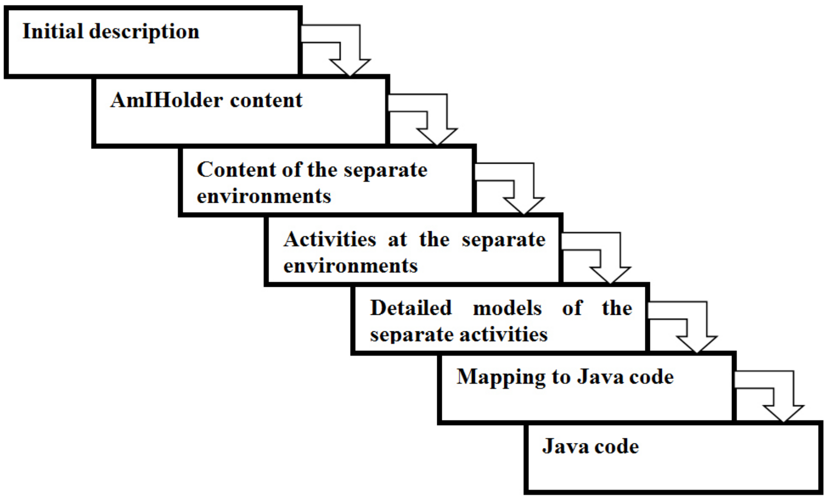

Initial description
AmIHolder content
Content of the separate environments
Activities at the separate environments
Detailed models of the separate activities
Mapping to Java code
Java code

Figure 14. Set of tasks included in the development of AmI-E.

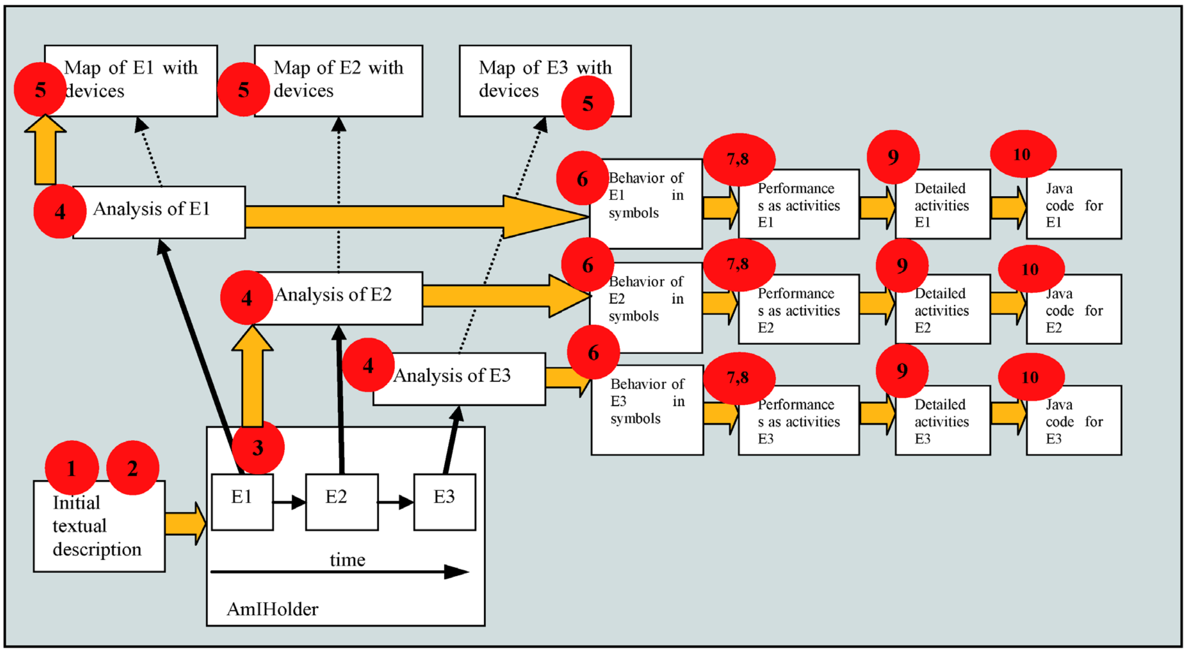


Figure 15. Entire flow of the AmI-E development process.

Semantics	Concrete Syntax
AmI Holder	▢ List of Environments
AmI-E	□ Container of the AmI comprising elements
Sequence	1, 2, 3…… →
Starting point	●
Finishing point	◉
Explanation Note	Finishing point

Figure 16. Concrete syntax of the AmI-Creator DSL for depicting of both: the environments at AmI Holder and overall performance at AmI Holder.

Every step provides the semantics (purpose or meaning) of the modelling stages and expresses the requirements to the modelling concepts to be implemented in the concrete syntax, one form of which refers to the graphical notations. Horizontal part specifies a set of the possible models and concrete modelling elements, participating in the development of AmI environments, which finalised with state-chart diagrams representing code. The modelling elements are abstracted in the groups, providing basic entities, from which the elements are derived and; models are constructed. The idea of the horizontal methodology with the appropriate modelling stages are summarised in Figure 14. According to it flowchart the entire flow should contain the stages expressed with the appropriate suggestion of concrete syntax:

- Initial description in textual form.
- AmIHolder content.
- Content of the separate environments.
- Activities at the separate environments and detailed models of the separate environments as graphical models.
- Code representation in Java language notations.

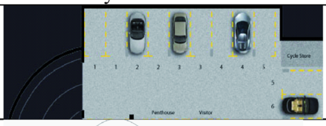
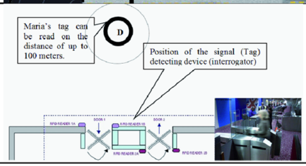
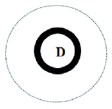
Semantic	Concrete syntax
Depiction of an environment	
Placement of the device in the environment (in this case the P-com device)	 Maria's tag can be read on the distance of up to 100 meters. D Position of the signal (Tag) detecting device (interrogator) RFID READER 1A DOOR 1 RFID READER 1B DOOR 2 RFID READER 2A RFID READER 2B
Device: The internal circle identified by the letter D refers to the device. The internal circle shows the diameter of the detection of the device.	
Detection of the device position	●
Explanation note	Display of parking place

Figure 17. Concrete syntax of AmI-Creator for coarse organization of the environments (the first, second and last row show the examples only).

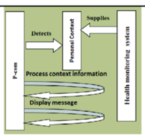
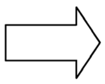
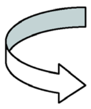
Semantics	**Concrete syntax**
Performance	 Detects Personal Context Supplies P-com Health monitoring system Process context information Display message
AmI entity	Name
Communication	
Self-communication	
Name of the communication	Generate/notify about

Figure 18. Concrete syntax of AmI-Creator for realization of the performances at the different environments.

These suggestions express the requirements to the Domain-specific language-AmI-Creator, which should provide means of textual, graphical and Java programming representations of the particular stages of the development process. To support the idea of MDD for the development of AmI-E it should be supported by the appropriate software development tools. The requirements to the tools comprise their capabilities of expressing of all necessary notations specified in DSL.

6.1. How A-Methodology Works

The A-methodology can be applied to the development of different domains with complex organisation in two ways.

Semantics	Concrete syntax
AmI-E	
Name of the environment	$\boxed{\text{Name}}$
Communication	●⟶
Self-communication	↩
Stereotype	**<<Environment>>**
AmI system entity	|
Description of the communication	**Send context through network to**
Pointing to the time of the activity	←·–·–·–·–·–·–·–

Figure19. Concrete syntax of the AmI-Creator DSL for depicting of more detailed performance at environments.

6.1.1. First Way

The first way is dedicated to the cases when a developer approaches to a development without previous knowledge about a domain and/or a way of its creation. The current research demonstrates a use of such way through a consideration of AmI domain. Such case requires studying the domain through the available sources of information, e.g published documentation. Consideration and analysis of the information should be followed by an extraction of the meaningful elements with their succeeding abstractions in groups, which represent the domain concepts accompanied with their intercommunications. The intercommunications of the entities can be purposeful and lead to a performing of some processes. The groups with their intercommunications and processes are represented in ontologies of real world (see A-MDA level M0). The ontologies represent the domain semantics and provide a list of meaningful elements for initial modelling to be undertaken at A-MDA level M1.

The purpose of the activities at the level M1 is to investigate the modelling stages forming the entire modelling process finalised with models of code. The result of the investigation specifies the entire set of activities to be completed towards the final target of modelling code. Every stage of the modelling process has its own aim, which requires its own expression in some kind of formal representation.

Such multi-level representations refer to notations of multi-language dedicated to AmI domain with its focusing on the development of AmI environments. The multi-language should have two main notations: textual and graphical. They correspond to the development stages (see **Figure 14**).

Concluding this part the current research shows that the first way is dedicated to preparing a base for the horizontal part of the methodology, which is devoted to the development of Am-E and described as a second way of using the methodology and dedicated to the development of AmI environments.

6.1.2. Second Way

The second way of using the methodology (also refer to continuing of the First way) should be used during the development of a prospective AmI-E, when a domain is already learned, tools are specified and set of models are already identified and complete. The activities comprising the Second way refer to the horizontal part of A-methodology (corresponding to **Figure 14**) and comprises:

1) Identification of the entire structure of the AmIHolder.
2) Identification of the AmI environment content.
3) Identification of the content of the AmIHolder, comprising the entire set of the environments and their sequence in order of their engagement in the entire process.

4) Clarification of the content of the separate environments with the specified performances.
5) Detailed consideration of the performances in a dedicated diagram.
6) Visualisation of the code of the final activities.

7. Evaluation through Validation, Verification and Testing

This part demonstrates the activities showing whether the provided approach feasible through validation, verification and testing. The validation was provided for ontology, the AmI-Creator's semantics and syntaxes, existing languages and tools. Verification is provided for the proposed hypothesis. Testing is provided for the entire approach.

7.1. Validation

The validation of the AmI ontology was provided on three different examples from different domains comprising business, health and self-care-related applications, corresponding to Maria Scenario[9], Virtual Hospital[10] and Tea-making application [63]. The validation refers to a checking whether the every entity of AmI ontology presents at the considered example.

Such, for example the Patient case shows that AmI activities take part at Environments (Reception room, Examination room, Surgeon room and Recovery room), which comprise AmIHolder. Every Environment contains a set of the Devices capable of detecting a different environmental data or Context. The example of such device is a device capable of a detection of a specific signal (or a Context) from a patient's tag. The Device is equipped with a data-detecting feature; the patient's tag is equipped with signal-generating features. Such features ensure the interconnections, which facilitate a Network. A signal's generating and its detection take Time. This validation showed the presence of all elements of AmI ontology: AmIHolder, Environment(s), Device(s), Context, Network(s) and Time. The result of the verification, summarized in the **Table 3**. It shows that all semantic concepts identified in AmI ontologies are found in all three examples, which proved that the obtained ontologies are valid properties of the AmI environments.

7.2. Verification

The verification of the considered existing languages and tools demonstrates that they cannot facilitate the development of AmI systems. This is because they cannot completely satisfy the requirements to building of AmI environments, narrowed to the certain number of notations and development steps. Such requirements refer to an ability of the tools to provide all notations necessary for building of the software application from AmI domain (**Table 4**). The table presents the comparisons of such tools as VP[11]-Visual-Paradigm (general purpose Software development tools) and MetaEdit+[12] (dedicated to the development of domain-specific languages). Such tools would realize a use of the proposed language which can depict the steps of the development of AmI-E comprising nine stages (depicted in the **Figure 15**). They comprise:

1) Initial textual description of the functionalities of the entire system and at the separate environments(at stage one).
2) Depiction the entire AmIHolder and environments in an architectural notations (at the stage two see the concrete syntax of AmI-Creator at **Figure 20**).
3) Expression of the analysis at the stages 1 and 2 in the modeling concepts, which are the graphical depictions of the AmIHolder comprising the Environments and their order of the participation in the overall perfor-

Table 3. Verification of AmI ontology.

Application	AmIHolder	Environment	Network	Device	Cotext	Time
Maria	Yes	Yes	Yes	Yes	Yes	Yes
Patient	Yes	Yes	Yes	Yes	Yes	Yes
Tea	Yes	Yes	Yes	Yes	Yes	Yes

[9]SCENARIOS FOR AMBIENT INTELLIGENCE IN 2010 final report at ftp://ftp.cordis.europa.eu/pub/ist/docs/istagscenarios2010.pdf
[10]link "Visit the Virtual Hospital" at http://content.dell.com/uk/en/healthcare/healthcare-solutions.aspx
[11]http://www.visual-paradigm.com/product/vpuml/
[12]http://www.metacase.com/resources.html

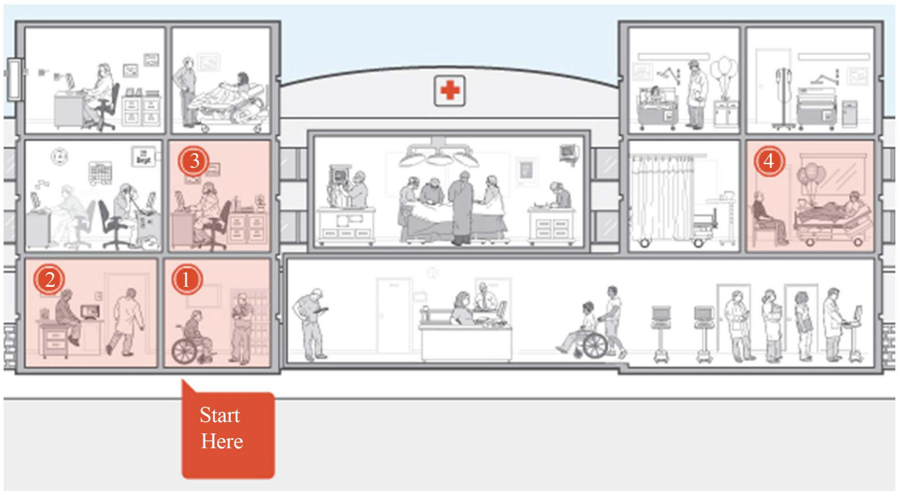


Figure 20. AmIHolder of the Patient case.

Table 4. Comparison of VP and MetaEdit+.

VP	MetaEdit+
Provide predefined graphical notations	Allows creation of graphical notations
Does not provide coherent development of an application	Does not provide coherent development of an application
Offers various diagrams meaning of which can be vague	Offers some pre-developed examples with graphical notations
Does not allow depiction of time and activities in the same diagram	No inbuilt examples showing time and activity representation
Generate Java code as code shells with	Instantly generate XML code
Attributes and methods and XML Schema	Requires a lot of efforts to develop code generation
Present separate diagrams only	Requires a lot of efforts to develop code generation
Generate Java shell from Class diagram only	Do not have means of the analysis of an application
Does not allow depiction of meta-concepts (which means it is not possible to reuse the developed parts)	Allows meta-concept; development profiles and reuse

mance (at the stage three see the concrete syntax of AmI-Creator at Figure 16).

4) Analysis of the activities at the separate environments (at the stage four see the concrete syntax of AmI-Creator at Figure 16).
5) Analysis of the kind of the devices and their placement at every environment (at stage five see the concrete syntax of AmI-Creator at Figure 17).
6) Analysis of the behavior of every environments (at stage six Figure 18).
7) Graphical representation of each performance allocated at the appropriate environment (at the stage seven see the concrete syntax of AmI-Creator at Figure 19).
8) Detailed representation of each performance (at the stage eight Figure 19).
9) Code representation (at the stage nine see the concrete syntax of AmI-Creator at page 14).

The Table 4 describes the features of VP and MetaEdit+, which are not fully satisfied the completion of the current task. The derived conclusion is based on the following speculations:

7.3. Validation of MetaEdit+ the DSL-Dedicated Tool

MetaEdit+ does not provide features of the analysis of the system under the development. The current research understands that MetaEdit+ is dedicated to a representation of very small systems (sometimes manually provide features of the code generation). Development of the large systems or representation of multi-level language introduces a problem of depicting of the links between the models. According to a MetaEdit+'s manual, a development of sophisticated language notations takes time.

MetaEdit+ supports meta-modeling features and advertises a full code generation. Closer look at the manual discovered that MetaEdit+ provides editor for a manual writing code. It provides a very limited (ready to use) transformation of graphical notations into XML. Taking in an account the content of the manuals, the current research's own experiments with the existing examples and attempts to develop an own multilevel language showed that a development of such language introduces a problem and requires an additional training (time on which will oversee the time allocated on the entire research). It also would require a lot of programming experience in writing a translator from graphic-based to a programming-based modeling (program-based implies a use of the conventional programming languages as Java, C++, C sharp).

7.4. Validation of Visual Paradigm-a General Software Development Tool

The overall UML at VP contains two types of diagrams: Structure (to depict statics in the modelling) and Behavior (to depict dynamics in the modeling). They comprise thirteen diagrams and eight sub-diagrams. The set of diagrams covers a very wide range of purposes for the development and at a first glance it looks like they possibly could solve the problem of modeling of AmI environments. The following text describes the results of the experiments with the existing tools. It shows how fully a use of UML can satisfy the requirements for the tools capable of holding all features necessary for the development of Ambient Intelligence environments.

The first step in the AmI-E development refers to the initial description of a prospective AmI-E. Visual Paradigm provides two types of the textual support, referring to the "textual analysis" and "report". The VP intention of using the "textual analysis" is to identify the future modeling elements, e.g. class, actor, activity, actions, whereas "report" features refers to a generation of the report from developed models. The second kind of the textual support can only be generated and for this reason it is not suitable for the initial description.

The first type of the textual support cannot be adapted for producing the initial documents to be discussed with the customer as it does not provide the features of importing an image file (with picture of the environment) or drawing a content of the environment for the third stage of the development. The activity of "Producing a high level of a graphical model of an entire system" by modeling features of VP shows that UML provides several types of diagrams capable of depictions different features and comprises Class diagram, Component diagram, Package diagram, Deployment diagram, Object diagram, Composite structure diagram.

The consideration of UML's class diagram shows the problems with depicting activities. For example Class diagram is not capable of depicting a notion of time, which is important for the current research context. Otherwise it is capable of the depiction of a Device, Environments with the allocated performances, which should be completed at the AmI-E.

As UML Use case diagram refers to a behavior diagram the current research tried to adapt it for a depiction of statics along with system functionalities. The Use Case diagram shows that a Device (expressed in an Actor UML notation) is a communicating device capable of communication with other devices. The Use case diagram can depict the set of operations (or performances) and places of their allocation, but cannot show the exact sequence of such; as well as cannot depict time when the operation is to take part.

The consideration of the other kinds of the diagrams comprising Component diagram, Package diagram, Deployment diagram, Object diagram, Composite structure diagram, Interaction overview diagram demonstrates that these diagrams are not dedicated to such features and their adoption would bring the same complications as lack of precise sequence of the operations and time depiction. The following experiment involves the consideration of UML's ability to cope with the tasks referring to the dynamic processes in the separate environments. For this purpose the current research considers the behavior diagrams: Use case, Activity, Interaction (Sequence, Communication, Interaction Overview, Timing) and State Machine diagrams.

Having considered the Use case diagram the research looks at the features of Activity diagram allowing a depiction of an Environment, the set of the elements comprising the environment and the performance.

The vertical swim-line of the Activity diagram is capable of an incorporation of the features representing environments. Despite activity diagram allows creation of an additional swim line dedicated to the depiction of time, the features of this diagram are not capable of the depiction of a precise point of time or length of time. The identification of the precise sequence of the operations courses a problem.

The consideration of Communication diagram showed that it can be adapted to a depiction of the AmI environments parts, but it is not capable of a depiction of the entire environment. The diagram is capable of a depiction of the sequence of operations in the graphical (arrowed) notations with the appropriate numbers. The diagram is capable of depicting of the overall process, but is not capable of an identification of a point or a length of the time in a form other than a textual note. There is also some confusion in the depiction of the sequence of the operations. For example in depiction of operations numbered as 2 (Context detection) and 3 (Performance) can show that the operation number 3 is depicted earlier then the operation number 2. This is incorrect: the performance (or reaction on signal detection) cannot take part earlier than the signal detection. The other point required mentioning is inability of an error correction: if a developer requires an inserting of an additional operation, he needs to start modeling since the very beginning as the tools would not display the corrected activities in the required order.

- In conclusion the research realizes that none of the tools are capable of depicting the entire AmI environments development process. Based on these considerations the research provides a solution presented as DSL in the meaning of multi-level language and taking some features of UML and MetaEdit+. Also based on the experiments with VP the current research proposes a solution: to provide a precise depiction of the time in the graphical notations supported with the features of expressions in numbers.

The verification of the Language approach to the development of AmI environments showed that the number of the obtained notations is sufficient for the development of all stages and all notations which were met in the considered examples.

7.5. Testing

The proposed solution was tested on the developments of three different applications comprising: Tea-making application, Maria-scenario and Patient at the surgery. Because of too large sizes the demonstration of two first applications are not included in this paper. The third application is presented fully. It demonstrates an entire process of using of A-methodology's horizontal part. The testing of the proposed solution comprises inspection of:

- Whether the proposed set of steps of the Horizontal methodology is sufficient for the development of AmI-E.
- Whether the proposed AmI-Creator is capable of depicting the all necessary entities.

7.6. The Full Example of the "Patient" Application Is as Follow

1) The proposed solution is demonstrated through the Patient example based on the step-by-step sequence of performances. The future application refers as a process-based. The area of the application of the system refers to health care organizations. The analysis comprises: "The purpose of the system under the development is providing an electronic maintenance of a patient records' assisting a person during his/her presence at a hospital. The treatment journey is starting at a patients' consultancy room and finishing at a patients' discharging room. The treatment is taking place at a sequence of different environments with different functionalities. All of these environments are supplied with a device containing patients' records. Completing the predefined functionalities requires a sequence of specified environments (patient reception room with nurse taking initial data from patient; patient examination room with doctor-consultant examining the patient health condition; surgery room with doctor providing operation; after surgery patient's room for patient recovery (visited by doctor-consultant and nurse)). The listed environments are supplied with equipment supported with sensing devices capable of detecting tags and voice commands. Such detections are based on hands free communications through wireless network. The communication requires time to complete a set of tasks allocated to the appropriate environments."
2) The area of the application of such system refers to health care organizations.
3) Analysis of the future system shows that it has to have a set of environments with contexts, detectable by the appropriate devices, devices network interconnections, time to commit an activity. The initial studying of the desired features showed that fulfillment of the features, servicing patient treatment at the hospital requires: devices (tags, interrogators, voice activity recognition, electronic device supplied with database as hospital server, computer-clients), context to be detected (e.g. tag signal or voice command), network features to send the information for storage and processing, time to complete request and response, environments the performances should take place. All of these refer to the properties of Ambient Intelligence, which support the purpose of maintenance of people everyday life-so such system is to be considered as of Ambient Intelligence.

4) AmIHolder content: “The AmIHolder of the Patient case consists of reception room, examination room, surgery room, recovery room” which are shown at the picture **Figure 20**.
5) The main elements of the future system, refer to environments are depicted in the diagram **Figure 21**, where every numbered environment corresponds the numbered environments at the **Figure 20**.
6) The analysis of the separate performances are as follow:

Reception room: On placing the patient at the appropriate position the patient’s tag is detected and the specified information is uploaded onto the tag.

Examination room: Room tag detection device detects the patient ID through the tag, identify the patient and displays the data on the device screen.

Surgeon (or operation theatre): Room tag detection device detects the patient ID through the tag, identify the patient, notifies surgeon about the patient health conditions and start recording the operation at the specified time.

Recovery room: Room tag detection device detects the patient ID through the tag, process the data, finds and displays the data about the patient.

The set of the environments at AmI Holder, organized in the order of their participation in the overall performance depicted in **Figure 21**.

7) The detailed description of the sub-performances in **Figure 22** comprises:

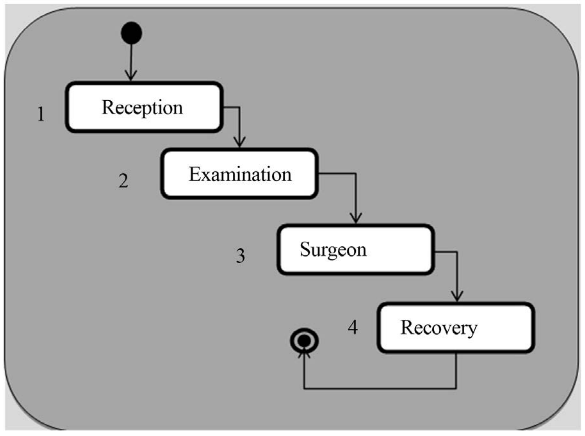


Figure 21. The environments at AmIHolder.

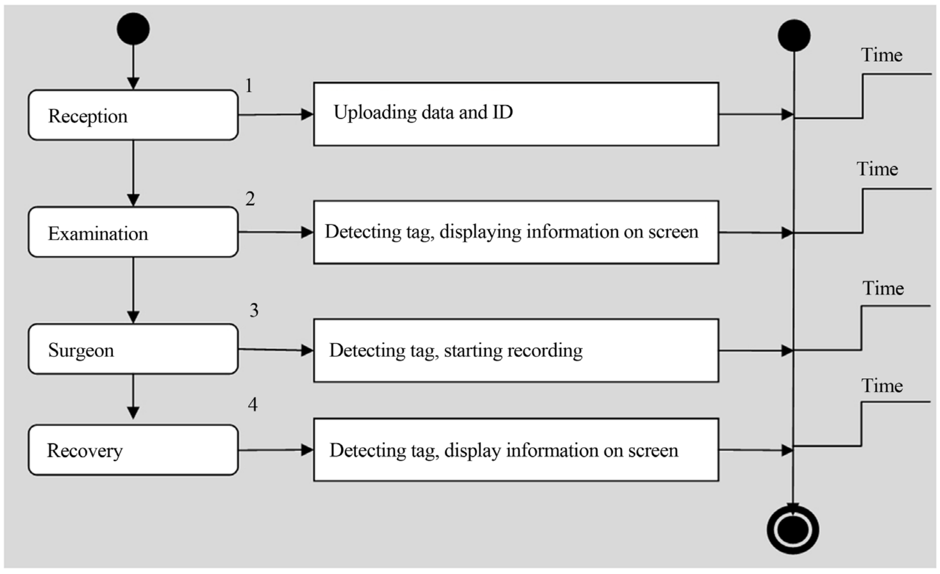

Reception
1
Uploading data and ID
Time
Examination
2
Detecting tag, displaying information on screen
Time
Surgeon
3
Detecting tag, starting recording
Time
Recovery
4
Detecting tag, display information on screen
Time

Figure 22. Analysis of the overall performance.

a) At the reception: The passive tag is placed at the allocated area; on the request communication it sends the responding signal, which makes the controlling device start uploading patient information if tag is empty.

b) At the examination room: the patient with the tag) is placed at the allocated area; the controlling device sends the request to communication; the tag responds by the signal; the controlling device ConDev) performs the appropriate activities.

c) At the operation theatre: the ConDev sends the request signal; the tag responds; the ConDev displays the information on a screen and stars recording the operation.

d) At the recovery room: the ConDev sends the request signal; the tag responds; the ConDev displays the information on a screen.

8) The identification of the needed devices are summarized in a dedicated **Table 5**. The identification of the needed devices comprise:

a) The patient's tag is invoked by a request to communication signal and responds to it. So this detectable-device refers to passive tag.

b) The requesting devices refer to controlling devices, situated at the controlling rooms of the appropriate environments.

9) Division of the overall performance into the separate performances is summarized in the **Table 6**.

10)The analysis of the sequence of the performances showed that they take part one after the other, which lead to conclusion that such AmI-E is a performance oriented.

11)The detail sequence of the sub-performances with the appropriate environments are as follow:

a) At the reception: The passive tag is placed at the allocated area; on the request communication it sends the responding signal, which makes the controlling device start uploading patient information if tag is empty **Figure 23**.

b) At the examination room: the patient (with the tag) is placed at the allocated area; the controlling device sends the request to communication; the tag responds by the signal; the controlling device (ConDev) performs the appropriate activities **Figure 24**.

c) At the operation theatre: the ConDev sends the request signal; the tag responds; the ConDev displays the information on a screen and stars recording the operation **Figure 25**.

d) At the recovery room: the ConDev sends the request signal; the tag responds; the ConDev displays the information on a screen **Figure 26**.

Table 5. The summary of the devices required for the Patient AmI-E.

Environment	Performance	Devices
Patient reception room	After the tag recognition the reception control device uploads a patients ID and data on the patient tag	Patient wrist tag refers to a passive tag (equipped with read-write RFID (Radio Frequency Identification) or SAW (Surface Acoustic Wave))
Doctor-consultant room, Operation theatre, patient recovery room	On the patient's taking an appropriate position the room control system Identifies the patient and display data on the appropriate screens	Patient wrists tags (e.g. read-write RFID (Radio Frequency Identification)) or SAW (Surface Acoustic Wave). ConDev device computer is equipped with interrogator.

Table 6. Division of the overall performance into the separate performances.

Performance	Performance description
Uploading ID and data about patient on his tag	On arriving to the hospital a patient is given a wrist electronic tag. When patient is registered, the information about him is stored on the server, holding database records of all patients. This is done wirelessly through the communication distance up to 10 meters: When patient is placed at the appropriate position (one at the time) the room controlling device detects the passive tag and sends the appropriate data on it
Identifying patient at the doctor-consultant room	When patient entered in the consultant's room, his passive tag is detected by client computer's interrogator; the room controlling device displays the information about the patient on a screen
Detecting patient at the operation theatre	When patient is in the operation theatre his tag is identified by the reader of the controlling device and displays the appropriate patient's record and start recording the operation
Detecting Patient at the recovery room	When patient entered in the recovery room, his active tag is detected by the controlling device, which finds and downloads patient's ID. When patient is discharged from the hospital his device is disposed

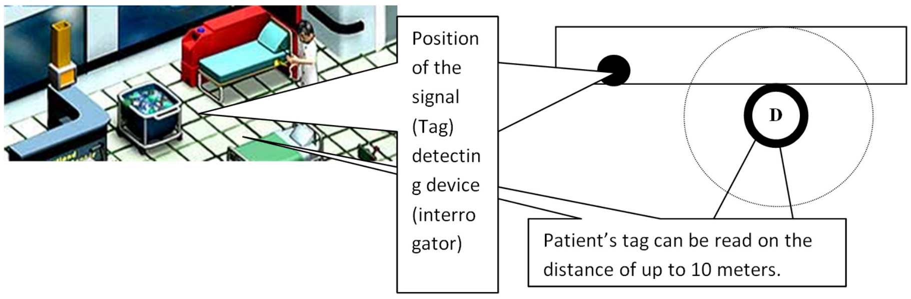

Position of the signal (Tag) detecting device (interrogator)
D
Patient's tag can be read on the distance of up to 10 meters.

Figure 23. Analysis of the performance at the reception.

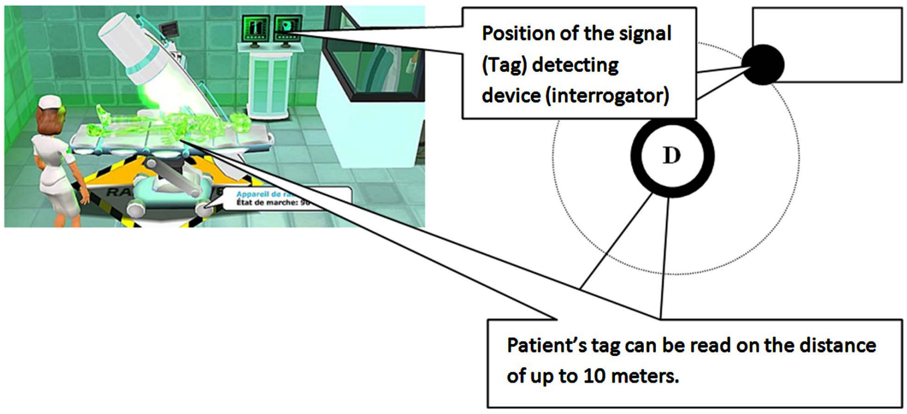

Position of the signal (Tag) detecting device (interrogator)
D
Patient's tag can be read on the distance of up to 10 meters.

Figure 24. Analysis of the performance at the examination room.

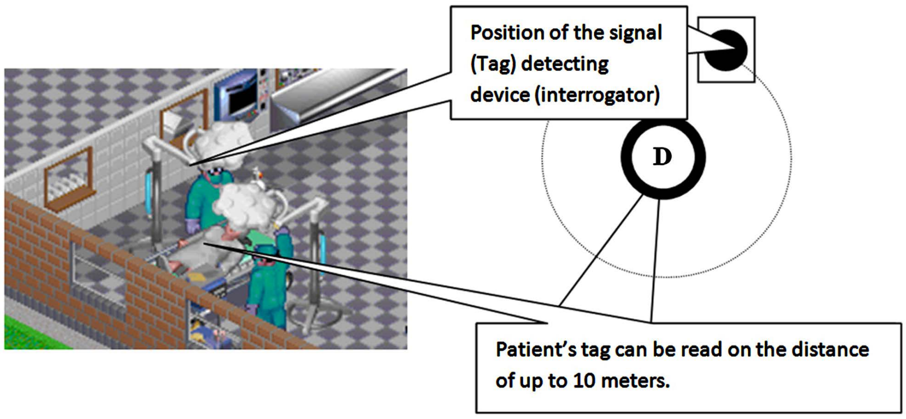

Position of the signal (Tag) detecting device (interrogator)
D
Patient's tag can be read on the distance of up to 10 meters.

Figure 25. Analysis of the performance at the operation theatre.

12) The modeling of the separate performances for example as modeling of the performances at the appropriate environments: Reception room (top right counter), Examination room (top left corner), Operation theatre (bottom right corner), Recovery room (bottom left corner); and obtaining a pattern is shown as in Figure 27.
13) More detailed analysis of the performances is shown in the Figure 28. They are the same for all of the environments. As it can be seen from the previous part both of the devices perform the following operations: "detect", "process" and "respond". In respect to this particular case the server side refers to Patient tagging device (or tag), whereas client part refers to the communication device. They perform the same operations, but the exact contents of the appropriate inference engine and network communications would depend on the discussion with the appropriate specialists about creating a knowledge base system (for an output inference) and a network adviser.
14) Mapping into Java class is shown in the Figure 29. This picture shows that for example class Reception (al-

Figure 26. Analysis of the performance at the recovery room.

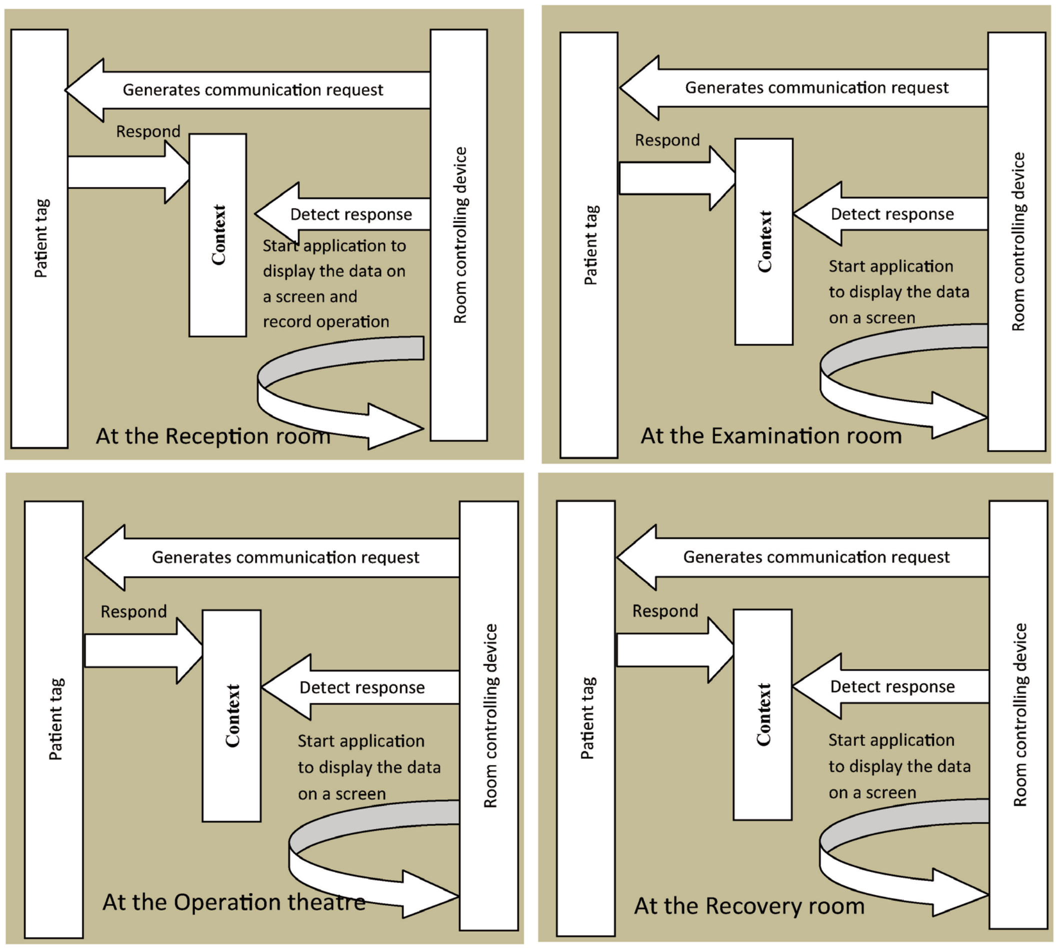

Generates communication request
Respond
Detect response
Start application to display the data on a screen and record operation
Patient tag
Context
Room controlling device
At the Reception room
Generates communication request
Respond
Detect response
Start application to display the data on a screen
Patient tag
Context
Room controlling device
At the Examination room
Generates communication request
Respond
Detect response
Start application to display the data on a screen
Patient tag
Context
Room controlling device
At the Operation theatre
Generates communication request
Respond
Detect response
Start application to display the data on a screen
Patient tag
Context
Room controlling device
At the Recovery room

Figure 27. Realization of the performances at the different environments through the modeling.

so Examination/Operation/Recovery rooms) is depicted at the diagram as a swim line "Environment". It corresponds to the Java class with the same corresponding name. Further (from the ontologies we learn that AmI environment has several properties) the properties of AmI environments map into the appropriate attributes of the class. Such swim lines depicted as Time/Context/PatientTag (or Device)/Network/ConDev (or Device) represent properties of the environments and map into the appropriate Java notations. The prop-

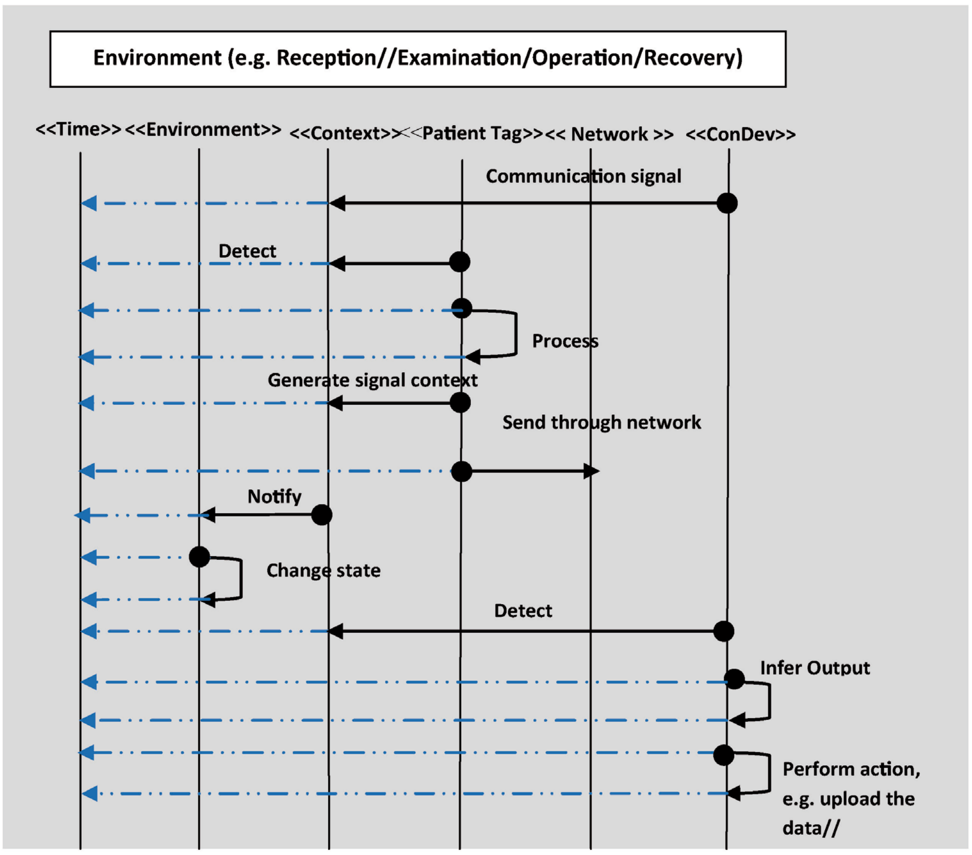

Environment (e.g. Reception//Examination/Operation/Recovery)
<<Time>> <<Environment>> <<Context>> <<Patient Tag>> << Network >> <<ConDev>>
Communication signal
Detect
Process
Generate signal context
Send through network
Notify
Change state
Detect
Infer Output
Perform action, e.g. upload the data//

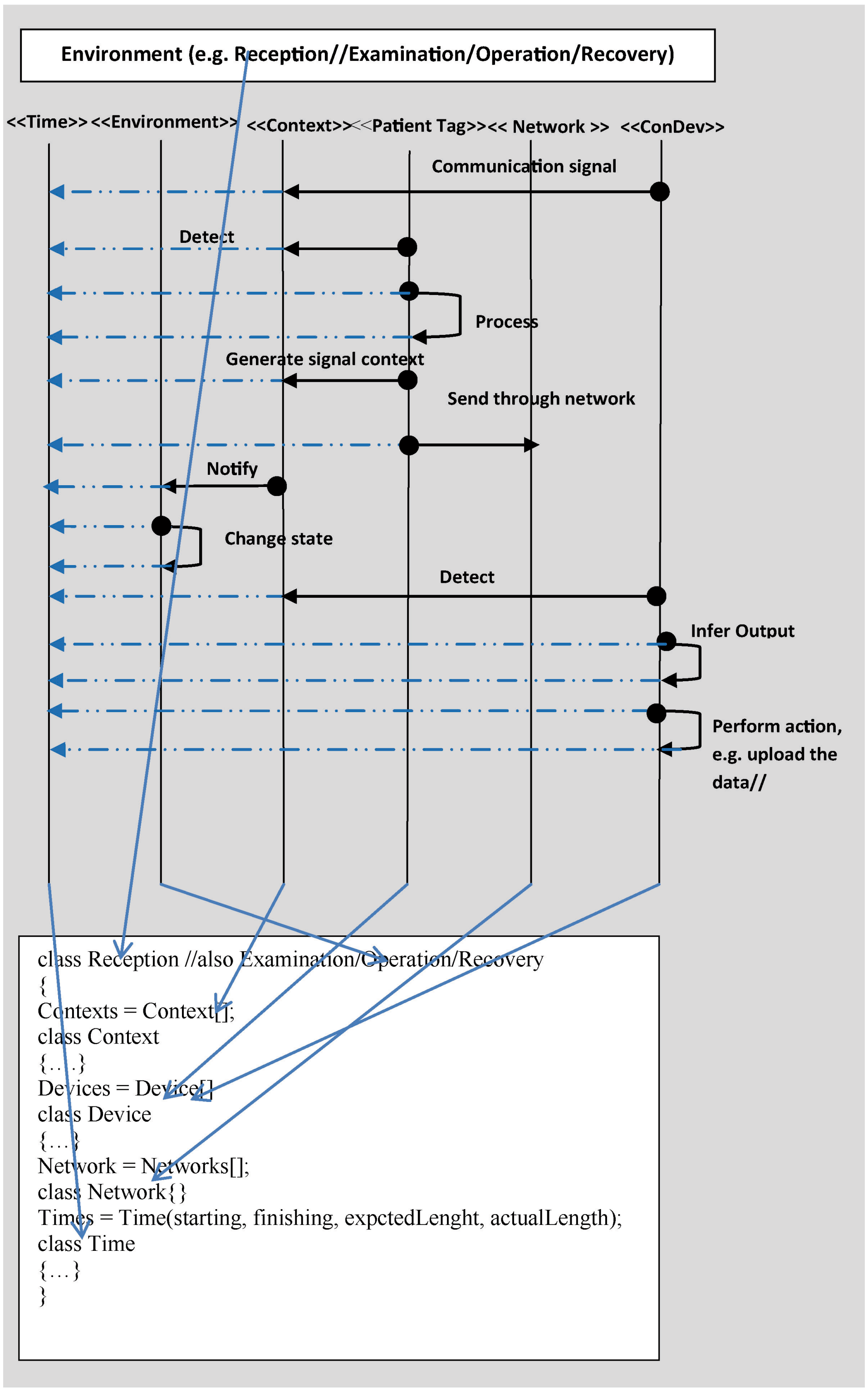
Figure 28. More detailed analysis of the performance at the reception.

erties are presented as classes because they have their own properties. The classes are shown as set of array data type because the instances of these classes can correspond to the several properties with different values. Such for example class Device can contain Patient Tag and Connecting Device. In this case the Device class will have two instances, what would be indicated as index of 2: Device [2].

8. Conclusions and Further Work

This part provides a summary of the completed research.

Along with the demonstration of how to develop an AmI-based software application from scratch the current research positively answered the research question "whether the attraction of SE and MDD into the development of AmI-E is the right approach to the standardization of such development". It also proved the research hypothesis that "MDD approach to the development of AmI environments is possible and can be produced". Giving a positive confirmation the current research points out on the benefits of using the proposed approach and expansion of its applicability to a development of other systems with complex organizations. The research showed a number of models derived during the process of the current research. These models can be used at the appropriate stages of development. Every model has its own notations, which refer to concrete syntax of AmI-Creator. The first part of the multi-level language is expressed in natural language. The second part takes the form of the notations depicted in Figure 16. The next part takes form of architectural sketch depicting the Environment, where the AmI activities are taking place (see Figure 23 and Figure 17). The expression of the next stage referred to the graphical notations as shown in the Figure 21 and Figure 16. Here a solid dot depicts a starting point of the AmI activities; boxes with the rounded corners depict the environments comprising an entire AmI holder. The arrowed lines show directions of the ordered AmI activities. The integers show the order of the environments participating in the process. The other concrete syntaxes of AmI-Creator are depicted at Figures 17-19. The mapping into the Java notations (which is part of AmI-Creator, borrowed from Java to finalize the task), is shown at Figure 29.



```
class Reception //also Examination/Operation/Recovery
{
Contexts = Context[];
class Context
{…..}
Devices = Device[]
class Device
{…}
Network = Networks[];
class Network{}
Times = Time(starting, finishing, expctedLenght, actualLength);
class Time
{…}
}
```

Figure 29. Mapping into Java classes.

To complete the presentation of the AmI-Creator one more set of notations is presented in **Figure 30**. Such notations are meant to be used, for example, for a modeling of Server/Client communications shown in **Figure 31**.

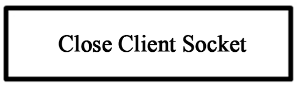
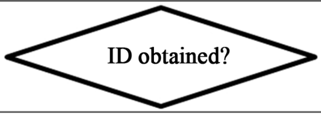
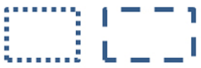
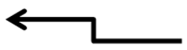
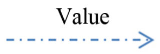
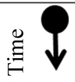
Semantics	Syntax
Activity	
Decision node	
Device edges	
Depicting an order of the events	
Time identification	
Time	

Figure 30. Concrete syntax of the AmI-Creator DSL for depicting of server client communications.

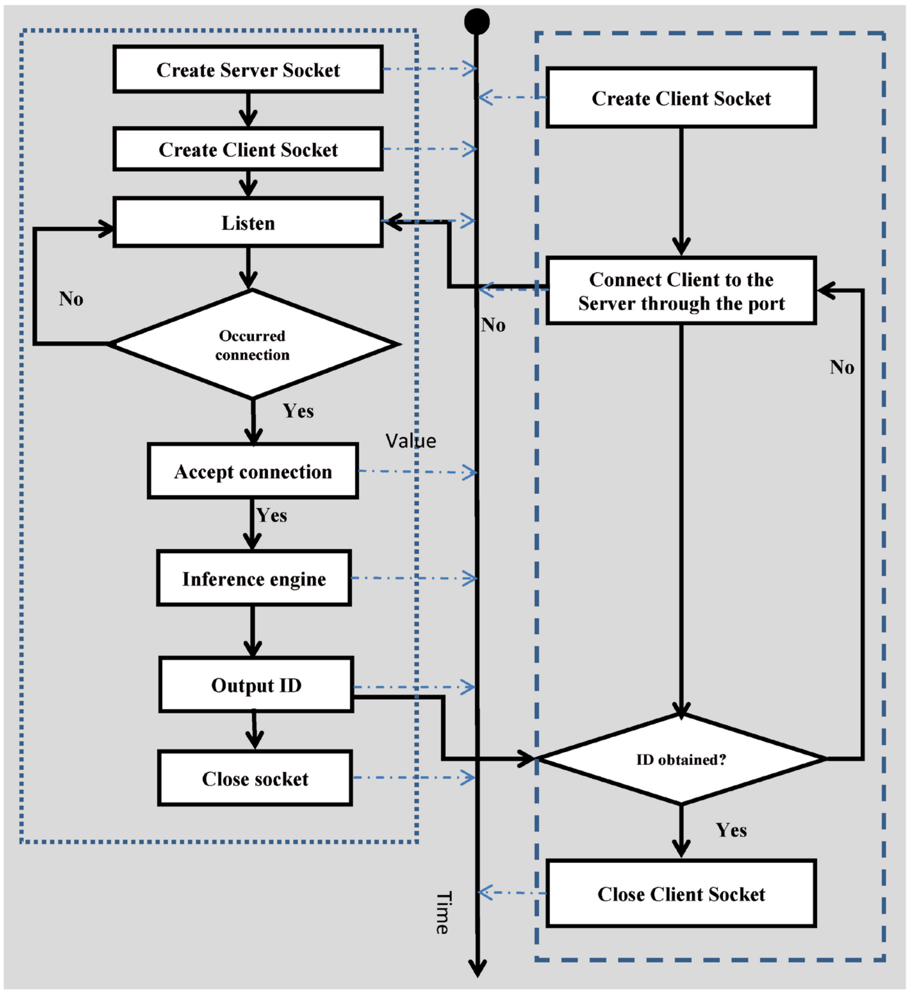


Figure 31. The example of modeling of server client communications.

8.1. Updates of the Existing Knowledge

In the author's opinion the current research produced the following knowledge updates at the fields of Ambient Intelligence and Software Engineering:

1) Adapted use of MDA.
2) New step-by-step methodology based on the specified sequence of models.
3) New insight into the development of DSL, starting from identification of modeling semantics, followed by the language syntax.
4) Development of AmI ontology providing semantics to the corresponding models AmI.
5) AmI ontology providing indistinguishable entities of Ambient Intelligence domain.
6) A methodology for the development of AmI environments finalized in code.
7) Visualization of the intercommunication devices.

8.2. Advantages of the AmI-Creator Approach in Comparisons with Other Latest Approaches

This section provides the demonstration of the advantages of the proposed approach through a comparative analysis of the two latest approaches published after the starting date of the current research.

The latest literature survey showed some of the AmI-relevant publications, dedicated to other areas of AmI e.g. towns and wider geographical areas, beach and dance floor. They do not produce DSL or AmI development methodologies. Some of them show stages such as identification of the purpose or development hardware and software (without giving the detailed explanations).

The first example proposes an approach to building a prototype. Alternative to De Ruyter's approach [55] the current research proposes a different insight into the problem of the development of AmI. It comprises planning and analyzing of the future system before building actual prototypes or full size model of the future environment. In the author's opinion such approach is based on modeling (showing planning ahead with identifying an initial purpose), choosing the appropriate devices (based on their ability to detect the context at the specified area), specifying the most appropriate placement in the environments; and providing coding—is a more advantageous way in the sense of saving efforts in the development of such systems. Such an approach also speeds up the coding process and reduces the end code errors. Checking a visual code model is always easier than checking a numerous lines of the actual code. The visualization of the development process through the symbolical representation, code visualization (through its depiction in graphical notations) can be discussed with the appropriate non-programmer specialist (as network or device engineer). Such discussion seeks for an immediate feedback.

The second example [64] also refers to the practice of a full size prototyping of AmI environments. It points to the importance of a system's deployment. It also points to a focus on small innovations on specific parts of home automation systems, which don't consider integration and deployment issues to present practical designs. The Zamora-Izquierdo's article offers "direct adaptation to specific cases using standard domotic technologies, for managing in-house devices. It also proposes an IP-based network for connecting the main home automation module with other platform elements". It refers to "A remote security system has been developed, and managing tasks are enabled via in-home control panels and an advanced 3D application for local and remote homeowner access." This was deployed on the prototype house testing domotic services. They point out the extension to e-health, elderly adaptation, green house automation.

Differently from this approach the current research does not specify a particular application area.

In contrast to both of the described approaches the current research pays particular attention to the provided feedback. Such feedback can help to speed up the process of the AmI-E development and avoid an unnecessary testing before planning and analyzing the prospective AmI-Es. Therefore, the Horizontal methodology defines the step-by-step process of the actual development of the prospective AmI-Es. Along with providing the means of intercommunication it provides the sequence of models, finalized by the models of the working code. Such modeling process leads to modeling of the code. Such approach can alleviate the process of coding and help to avoid coding errors. It also can help to speed up the entire process for the development of AmI-E.

The development of the Patient AmI-E took sixteen single person's working hours of to fully develop such environment (by hand) and model the final code. In comparison with the information given in the De Ruyter [55] article the process of mock-upping the similar environment (e.g. business office) takes several years and a considerable amount of the materials and supporting team.

By producing A-methodology the current research proved that SE MDD (showing way from one model to the other) is possible. The presented AmI-Creator with the appropriate tested methodology and demonstration of the developed environments proved that MDD development exists.

8.3. State-of-Art of the Research

The current research can be considered as state-of-art in the chosen field, which is dedicated to serving needs of the third wave of computing (AmI as Ubicomp) by means of fourth generation of software development (showing a way to code generation through a realization of mapping).

The reason for a considering the presented work as the state-of-art is because it presents the easiest way of creating of Ambient Intelligence environment from scratch. The benefits of using such approach cover leaving coding to the latest stage of the development. This approach helps to reduce considerably (or completely avoid) error in the final code. Providing these, the presented work helps to reduce labor and time consumption. It also allows producing the AmI environments in easy and error-reducing steps.

Furthermore, the research provides the direction for the further development in AmI domain by extension of its applicability to the other frameworks (for example for concurrency). The current research presents a process-oriented framework. Moreover, the current research assumes the applicability (or adaptation) of the methodology to other systems with a complex organization. Besides, the current research based its own realization of the subject on the consideration of the existing SE approaches and AmI-dedicated experiences (shown appropriately in **Figure 32**).

8.4. Further Directions

The realizations and completion of the identified subject depicted in **Figure 32** as "Existing information from world knowledge" was provided through the research of an AmI development approach, which also referred to a software development approach (approach to development of DSL through initial identification of the language semantics and A-methodology comprising Vertical and Horizontal parts)—referred to the Box named "Targeting knowledge". The new areas for the further development and obtaining a new knowledge (depicted as "New areas for the research based on the obtained knowledge") arise from the "Targeting knowledge" of the current

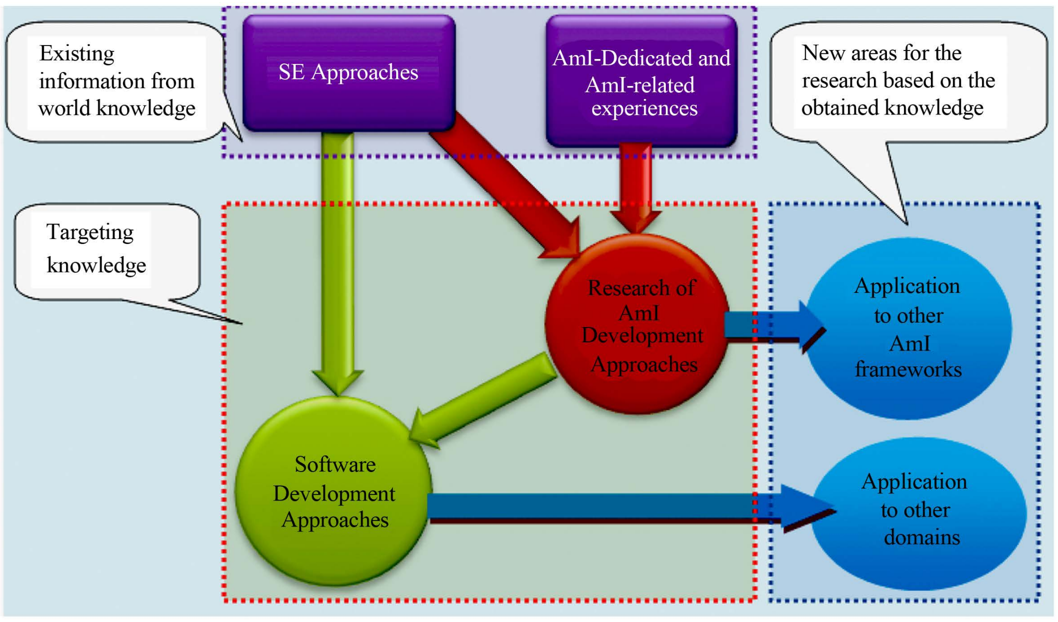

Existing information from world knowledge
SE Approaches
AmI-Dedicated and AmI-related experiences
New areas for the research based on the obtained knowledge
Targeting knowledge
Research of AmI Development Approaches
Application to other AmI frameworks
Software Development Approaches
Application to other domains

Figure 32. Direction for the further development in AmI domain by extension of the methodology to the other frameworks of AmI domain; and applicability (or adaptation) of the methodology to other systems with complex organization.

research. The "Research of AmI development Approach" can be extended to the development of other AmI domain frameworks (e.g. collaborations of several sensing devices leading to single or multiple decision output system, based on different sensing readings). The "Software development approaches" (referred to A-methodology) can be applied to other areas of systems with complex organization.

Acknowledgements

I would like to acknowledge EPSRC funding for making this research happened.

References

[1] Peinado, I., Arredondo, M.T., Villalba, E., Salvi, D. and Ottaviano, M. (2009) Patient Interaction in Homecare Systems to Treat Cardiovascular Diseases in the Long Term. *EMBC*: 2009 *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Minneapolis, 3-6 September 2009, 308-311.

[2] Gondal, I. (2007) Ambient Cardiac Expert: A Cardiac Patient Monitoring System Using Genetic and Clinical Knowledge Fusion. 6th *IEEE*/*ACIS International Conference on Computer and Information Science*, Melbourne, 11-13 July 2007, 496-501.

[3] Vales-Alonso, J., López-Matencio, P., Gonzalez-Castaño, F.J., *et al*. (2010) Ambient Intelligence Systems for Personalized Sport Training. *Sensors*, **10**, 2359-2385. http://dx.doi.org/10.3390/s100302359

[4] Bravo, J., Hervás, R., Casero, G., Peña, R., Vergara, M., Nava, S.W., Chavira, G. and Villarreal, V. (2008) Enabling NFC Technology to Public Services. 1*st International Ambient Intelligence Forum* 2008, Hradec Králové, 15-16 October 2008, 58-65.

[5] Garcia, O., Tapia, D.I., Rodríguez, S. and Corchado, J.M. (2010) Ambient Intelligence Application Scenario for Collaborative E-Learning. 23*rd International Conference on Industrial*, *Engineering and Other Applications of Applied Intelligent Systems*, Cordoba, 1-4 June 2010, 407-416.

[6] Norgall, T. (2009) Fit and Independent in the Aging Population Using Technology. From Concept to Reality. *Bundesgesundheitsblatt-Gesundheitsforschung-Gesundheitsschutz*, **52**, 297-305. http://dx.doi.org/10.1007/s00103-009-0789-5

[7] Streitz, N. (2007) Designing for People in Ambient Intelligence Environments. *International Conference on Ambient Intelligence Developments*, Ami.d 2007, Springer, Berlin, 47-54.

[8] Remagnino, P., Foresti, G.L. and Ellis, T. (2005) Ambient Intelligence: A Novel Paradigm. Springer, Berlin. http://dx.doi.org/10.1007/b100343

[9] Feng, L., Apers, P.M.G. and Jonker, W. (2004) Towards Context-Aware Data Management for Ambient Intelligence. 15*th International Conference on Database and Expert Systems Applications*, **3180**, 422-431.

[10] Aarts, E. (2003) 365 Days Ambient Intelligence in Homelab Ambient Intelligence: Building the Vision. Royal Philips Electronics.

[11] Weiser, M. and Brown, J.S. (1996) The Coming Age of Calm Technology. http://www.johnseelybrown.com/calmtech.pdf

[12] Ali, R., Abdel-Naby, S., Maña, A., Muñoz, A. and Giorgini, P. (2008) Agent Oriented Ami Engineering. *Developing Ambient Intelligence*, Springer, Paris, 166-179.

[13] Stuikys, V. and Damasevieius, R. (2007) Variability-Oriented Embedded Component Design for Ambient Intelligence. *Information Technology and Control*, **36**, 16-29.

[14] Augusto, J. (2010) Past, Present and Future of Ambient Intelligence and Smart Environments. In: *Agents and Artificial Intelligence*, Vol. 6, Springer, Berlin, 3-15.

[15] Eisenhauer, M., *et al*. (2009) Ambient Healthcare Systems Using the Hydra Embedded Middleware for Implementing an Ambient Disease Management System. 2*nd International Conference on Health Informatics* (Healthinf 2009), Porto, 14-17 January 2009, 82-89.

[16] Abascal, J., Bonail, B., Gardeazabal, L., Lafuente, A. and Salvador, Z. (2009) Managing Intelligent Services for People with Disabilities and Elderly People. 5*th International Conference on Universal Access in Human-Computer Interaction Held at the HCI International* 2009, San Diego, 19-24 July 2009, 623-630.

[17] Carneiro, D., Costa, R., Novais, P., Neves, J., Machado, J. and Neves, J. (2008) Simulating and Monitoring Ambient Assisted Living. *European Simulation and Modelling Conference*, Le Havre, 27-29 October 2008, 175-182

[18] De Paz, J.F., Rodriguez, S., Sánchez, J.M., de Luis, A. and Corchado, J.M. (2007) Context Aware Hybrid Agents on Automated Dynamic Environments. 2*nd International Workshop on Hybrid Artificial Intelligence Systems*, Salamanca, 12-13 November 2007, 25-32.

[19] Bonino, D. and Corno, F. (2008) Dogont-Ontology Modeling for Intelligent Domotic Environments. 7*th International*

Semantic Web Conference (*ISWC* 2008), Karlsruhe, 26-30 October 2008, 790.

[20] Cilla, R., Patricio, M.A., Belanga, A. and Molina, J.M. (2009) Non-Supervised Discovering of User Activities in Visual Sensor Networks for Ambient Intelligence Applications. 2*nd International Symposium on Applied Sciences in Biomedical and Communication Technologies* 2009, Bratislava, 24-27 November 2009, 244-249.

[21] Gabel, O., Litz, L. and Reif, M. (2005) NCS Testbed for Ambient Intelligence. *IEEE International Conference on Systems*, *Man and Cybernetics*, **1**, 115-120.

[22] Borrego-Jaraba, F. (2010) NFC Solution for the Development of Smart Scenarios Supporting Tourism Applications and Surfing in Urban Environments. *Trends in Applied Intelligent Systems*. *Lecture Notes in Computer Science*, **6098**, 229-238.

[23] Christie, P., Ng, R.K.M., Doornbos, G., Heringa, A., Kumar, A. and Nguyen, V.H. (2006) Xtreme Low Power Technlogy Development Using a Virtual Design Ow: Enabling Technologies for Ambient Intelligence Applications. In: Mukherjee, S., Aarts, R.M., Roovers, R., Widdershoven, F. and Ouwerkerk, M., Eds., *Amiware Hardware Technology Drivers of Ambient Intelligence*, *Philips Research*, Vol. 5, Springer, Dordrecht, 245-261.

[24] Filipe, P.P., Morgado, L. and Mamede, N.J. (2007) An Adaptive Domain Knowledge Manager for Dialogue Systems. *ICEIS*, **5**, 45-52.
http://www.researchgate.net/publication/220709594_An_Adaptive_Domain_Knowledge_Manager_for_Dialogue_Systems/file/9fcfd51373c052b389.pdf

[25] Gonzalez, E., *et al*. (2009) Atlintida: A Robust Indoor Ultrasound Location System: Design and Evaluation. 3*rd Symposium of Ubiquitous Computing and Ambient Intelligence* 2008, *Advances in Soft Computing*, **51**, 180-190.

[26] Kartakis, E. and Stephanidis (2010) Research Center Amitukaiserslauten. Techical Report, University of Kaiserslautern, Kaiserslautern.

[27] Seremeti, L. (2010) Ontology-Based Representation of Activity Spheres in Ubiquitous Computing Spaces. *International Joint Conference on Computer*, *Information*, *Systems Sciences and Engineering*, Springer, Berlin, 439-443.

[28] Chen, R., Hou, Y.B., Huang, Z.Q. and He, J. (2009) Modeling the Ambient Intelligence Application System: Concept, Software, Data, and Network. *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews*, **39**, 299-314. http://dx.doi.org/10.1109/TSMCC.2009.2014390

[29] Grammenos, D. and Georgalis, Y. (2009) Rapid Prototyping of an Ami-Augmented Office Environment Demonstrator. *Human-Computer Interaction*, *Ubiquitous and Intelligent Interaction*, **5612**, 397-406.

[30] Fuentes, L. and Gamez, N. (2008) Modeling the Context-Awareness Service in an Aspect-Oriented Middleware for AmI. 3*rd Symposium of Ubiquitous Computing and Ambient Intelligence*, **51**, 159-167.

[31] Lopezde Ipina, D., *et al*. (2005) A Middleware for the Deployment of Ambient Intelligent Spaces. *Workshop of Ambient Intelligence in Everyday Life*, **3864**, 239-255.

[32] Abascal, J., de Castro, I.F., Lafuente, A. and Cia, J.M. (2008) Adaptive Interfaces for Supportive Ambient Intelligence Environments. *Computers Helping People with Special Needs*. *Lecture Notes in Computer Science*, **5105**, 30-37.

[33] Abril-Jimenez, P. (2009) Design Framework for Ambient Assisted Living Platforms. *Universal Access in Human-Computer Interaction*. *Intelligent and Ubiquitous Interaction Environments*, *Lecture Notes in Computer Science*, **5615**, 139-142.

[34] Ali, R. (2008) Agent Oriented Ami Engineering. *Developing Ambient Intelligence*, *Proceedings of the International Conference on Ambient Intelligence Developments*, Springer Verlag, Paris, 166-179.

[35] Kaasinen, E., Niemelä, M., Tuomisto, T., Välkkynen, P., Jantunen, I., Sierra, J., Santiago, M.A. and Kaaja, H. (2010) Ubimedia Based on Readable and Writable Memory Tags. *Multimedia Systems*, **16**, 57-74.
http://dx.doi.org/10.1007/s00530-009-0171-3

[36] Arroyo, R.F., Garrido, J.L., Gea, M. and Haya, P.A. (2006) A Design Model Applied to Development of Ami Systems. *International Conference on Ubiquitous Computing*, California, 17-21 September 2006.
http://ceur-ws.org/Vol-208/paper13.pdf

[37] Shirehjini, A.A. (2005) 3dsim: Rapid Prototyping Ambient Intelligence. *sOc-EUSAI* '05 2005 *Joint Conference on Smart Objects and Ambient Intelligence*: *Innovative Context-Aware Services*: *Usages and Technologies*, Grenoble, 303-307. http://doi.acm.org/10.1145/1107548.1107621

[38] Imanol Durán, J., Cobelo, J. and Laka, J. (2008) A Product Line Approach for Ami Environments. *AmI* 2008, Springer, Berlin, 63-69.

[39] Bravo, J., Alamán, X. and Riesgo, T. (2006) Ubiquitous Computing and Ambient Intelligence, New Challenges for Computing. *Journal of Universal Computer Science*. *Special Issue*, **12**, 233-235.

[40] Silva, J.L., Campos, J.C. and Harrison, M.D. (2009) An Infrastructure for Experience Centered Agile Prototyping of Ambient Intelligence. *Proceedings of the* 1*st ACM SIGCHI Symposium on Engineering Interactive Computing System*, EICS 2009, Pittsburgh, 15-17 July 2009, 79-84.

[41] Seremeti, L. and Kameas, A. (2010) Ontology-Based Representation of Activity Spheres in Ubiquitous Computing Spaces. *Innovations and Advances in Computer Sciences and Engineering*, Springer, Berlin, 439-443.

[42] Munoz, A., *et al*. (2008) Model Checking Ambient Intelligence with Avispa. 1*st International Ambient Intelligence Forum* 2008, 182-193.

[43] Kaiserslautern, E. (2009) Research Center Amitukaiserslauten. Technical Reports, University of Kaiserslautern, Kaiserslautern.

[44] Nehmer, J., Becker, M., Karshmer, A. and Lamm, R. (2006) Living Assistance Systems: An Ambient Intelligence Approach. *Proceedings of the* 28*th International Conference on Software Engineering*, *ICSE* '06, ACM, New York, 43-50.

[45] Gasevic, D. (2006) Model Driven Architecture and Ontology Development. Springer, Berlin.

[46] Frankel, S. (2003) Model-Driven Architecture Applying MDA to Enterprise Computing. Wiley, Hoboken.

[47] Mellor, S.J., Kendall, S., Uhl, A. and Weise, D. (2004) MDA Distileed. Addison Wesley, Boston.

[48] MetaCase (2011) Introduction to Domain-Specific Modeling. http://www.metacase.com/resources.html

[49] Cernosek, G. (2004) The Value of Modeling. IBM Software Group.
http://www.ibm.com/developerworks/rational/library/6007.html

[50] Emiliani, P.L., Billi, M., Burzagli, L. and Gabbanini, F. (2008) Design for all in the Ambient Intelligence Environment. *Computers Helping People with Special Needs*, **5105**, 123-129.

[51] Stephanidis, C., et al. (2007) Universal Access Issues in an Ambient Intelligence Research Facility. Universal Access in Human-Computer Interaction: Ambient Interaction. Pt 2, *Proceedings of* 4*th International Conference on Universal Access in Human-Computer Interaction*, Vol. 4555, Beijing, 22-27 July 2007.
http://dx.doi.org/10.1007/978-3-540-73281-5

[52] Bosse, T., Castelfranchi, C., Neerincx, M., Sadri, F. and Treur, J. (2008) First International Workshop on Human Aspects in Ambient Intelligence: Preface. In: Mhlhuser, M., Ferscha, A. and Aitenbichler, E., Eds., *Constructing Ambient Intelligence*. *Communications in Computer and Information Science*, Vol. 11, Springer Berlin Heidelberg, 261.

[53] Augusto, J. (2008) Ambient Intelligence: Basic Concepts and Applications. In: Filipe, J., Shishkov, B. and Helfert, M., Eds., *Software and Data Technologies*. *Communications in Computer and Information Science*, Vol. 10, Springer Berlin Heidelberg, 16-26.

[54] David, B., Chalon, R., Delotte, O. and Masserey, G. (2007) Orchestra: Formalism to Express Static and Dynamic Model of Mobile Collaborative Activities and Associated Patterns. *Human-Computer Interaction*, Pt 1, *Interaction Design and Usability*, *Lecture Notes in Computer Science*, **4550**, 1082-1091.

[55] De Ruyter, B., van Loenen, E. and Teeven, V. (2007) User Centered Research in ExperienceLab. Ambient Intelligence, *Lecture Notes in Computer Science*, **4794**, 305-313.

[56] Hadzic, M. and Chang, E. (2010) Application of Digital Ecosystem Design Methodology within the Health Domain. *IEEE Transactions on Systems*, *Man and Cybernetics*, *Part A*: *Systems and Humans*, **40**, 779-788.

[57] Hadzic, M. and Dillon, T.S. (2008) Application of Digital Ecosystems in Health Domain. 2*nd International Conference on Digital Ecosystems and Technology*, Phitsanulok, 26-29 February 2008, 543-547.

[58] Consel, C. (1998) Architecturing Software Using a Methodology for Language Development. 10*th International Symposium on Programming Language Implementation and Logic Programming*, *Lecture Notes in Computer Science*, **1490**, 170-194.

[59] van Deursen, A., Klint, P. and Visser, J. (2000) Domain-Specific Languages: An Annotated Bibliography. *SIGPLAN Notices*, **35**, 26-36. http://doi.acm.org/10.1145/352029.352035

[60] Chandrasekaran, K., Ramya, I.R. and Syama, R. (2009) Ambi Graph: Modeling Ambient Intelligent System. *International Conference on Computer Engineering and Technology*, **1**, 441-445.

[61] Ali, R., Abdel-Naby, S., Maña, A., Muñoz, A. and Giorgini, P. (2008) Agent Oriented Ami Engineering. *Developing Ambient Intelligence*, *Proceedings of the International Conference on Ambient Intelligence Developments*, Springer Vergas, Paris, 166-179.

[62] Confluence Admin (2007) Ontologies Home. http://openehr.org/wiki/display/ontol/Ontologies+Home

[63] Olivier, P., Xu, G.Y., Monk, A. and Hoey, J. (2009) Ambient kitchen: Designing Situated Services Using a High Fidelity Prototyping Environment. *Petra* '09, 2*nd International Conference on Pervasive Technologies Related to Assistive Environments*, Corfu, 9-13 June 2009, 1-7.

[64] Zamora-Izquierdo, M.A., Santa, J. and Gomez-Skarmeta, A.F. (2010) An Integral and Networked Home Automation Solution for Indoor Ambient Intelligence. *IEEE Pervasive Computing*, **9**, 66-77.
http://dx.doi.org/10.1109/MPRV.2010.20