5.
Reuse-Oriented Approach for Incremental and Systematic Development of Intentional Ubiquitous Applications

> *"Users will be able to choose from among a variety of devices to gain mobile, high-bandwidth access to data and computational resources anywhere on the network. These devices will be intuitive, attractive and responsive. They will automatically adapt their behavior to suit the current user and context".*
>
> *Roy Want, Bill N. Schilit, Norman I. Adams, Rich Gold, Karin Petersen, David Goldberg, John R. Ellis and Mark Weiser, "The ParcTab Ubiquitous Computing Experiment,"Journal of Mobile Computing, pp. 45-102, 1996.*

In this Chapter we describe our reuse-oriented approach. First, we present an overview of our approach by emphasizing the *Domain Engineering of Ubiquitous Applications – i.e. Development for Reuse* – and the *Ubiquitous Application Engineering* – i.e. *Development with Reuse*. We consider for both developments their entries, controls, mechanisms and exits by using the **S**tructured **A**nalysis and **D**esign **T**echnique (SADT) (Marca and McGowan 1987). It is important to remember that this Chapter depends on the previous one, in which we describe the *Domain Engineering of Ubiquitous Applications* – performed by us during the last four years, from 2007 to 2010 – by proving building blocks.

Figure 5.1 illustrates the proposed process, which has two main steps: (1) *Domain Engineering of Ubiquitous Applications*, and (2) *Ubiquitous Application Engineering*. First, in the *Domain Engineering of Ubiquitous Applications*, presented in Chapter 4, the *Software Engineers* developed the artifacts (i.e. REUSE-ORIENTED BUILDING BLOCKS – also discussed in the previous Chapter). This activity is centered on *Ubiquitous Issues* (e.g. device and user heterogeneity, mobility, distributed environments, content adaptability need, and others), and oriented by the *Investigation of Different Ubiquitous Applications*, *Ubiquitous Computing Literature*, and GENERIC CONTROLS (e.g. from the *Conceptual Model of the i* Framework* to the *Conceptual Model of the*

Persistence Framework). This engineering also involves interactions with *Users*. Moreover, TOOLS are mechanisms that help the software engineers in this activity.

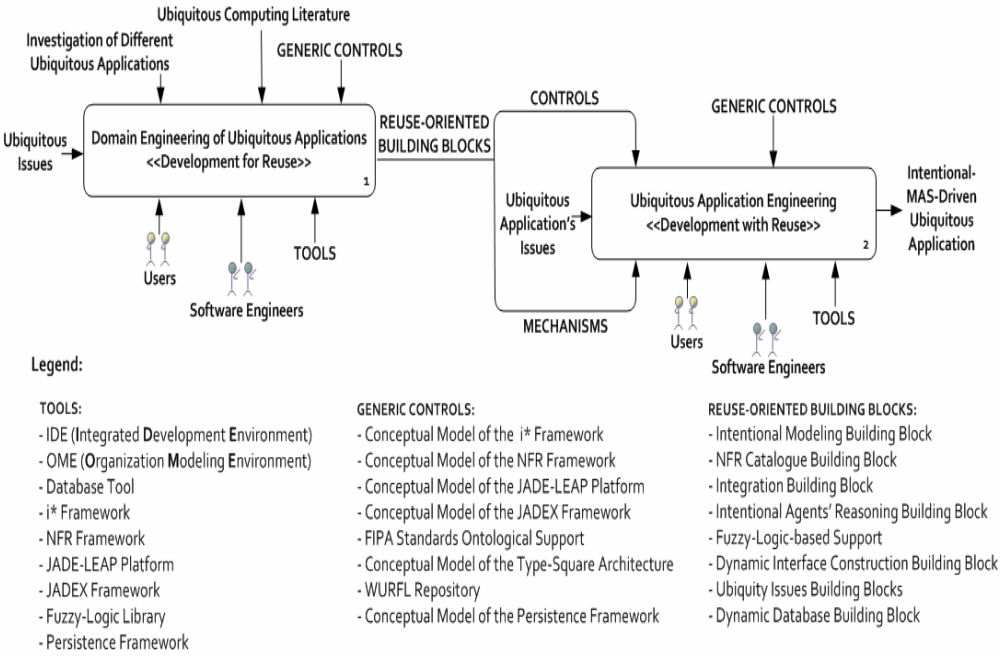

Figure 5.1 - Development of intentional-MAS-driven ubiquitous applications based on a reuse-based approach

In the *Ubiquitous Application Engineering*, the *Software Engineers*, centered on *Ubiquitous Application's Issues* (e.g. specific privacy policies, specific business rules, specific non-functional requirements and others) systematically develop the *Intentional-MAS-Driven Ubiquitous Application*. The REUSE-ORIENTED BUILDING BLOCKS' **CONTROLS** (i.e. conceptual models of the building blocks produced in the *Domain Engineering of Ubiquitous Applications*) and the GENERIC CONTROLS orient the software engineers in the application's development in accordance with the *Users*. The computational support is provided by TOOLS and REUSE-ORIENTED BUILDING BLOCKS' **MECHANISMS** (i.e. frameworks, libraries, catalogues and patterns produced in the *Domain Engineering of Ubiquitous Applications* to facilitate the development in the *Ubiquitous Application Engineering*).

In Section 5.1, we present the use of REUSE-ORIENTED BUILDING BLOCKS in the *Ubiquitous Application Engineering*. It is relevant to emphasize that the REUSE-ORIENTED BUILDING BLOCKS' CONTROLS represent conceptual models that among other contributions describe the way-of-working

based on the building blocks, their knowledge and concepts. The REUSE-ORIENTED BUILDING BLOCKS' MECHANISMS represent frameworks, libraries, catalogues and patterns that can be used as they are provided or reused by instantiation or extension to better satisfy the application's needs in its systematic development. Furthermore, we zoom in the proposed reuse-oriented architecture (Section 5.2) and the life-cycle used to incrementally and systematically develop intentional-MAS-driven ubiquitous applications (Section 5.3). Finally, Section 5.4 presents some final remarks.

5.1. Working with Reuse-Oriented Building Blocks

As mentioned, in the *Ubiquitous Application Engineering* the focus is on the development with reuse by directly reusing the building blocks as well as by extending or instantiating them. Only to present some details in this field, we illustrate – from Section 5.1.1 to Section 5.1.8 – the usage or instantiation or extension of some building blocks in the *Ubiquitous Application Engineering*.

5.1.1. Intentional Modeling Building Block Reuse

In order to model ubiquitous applications by using distributed intentionality, we suggest the reuse-oriented support centered on the i* Framework. In this field the main REUSE-ORIENTED BUILDING BLOCK is:

- The *Intentional Modeling Building Block* that offers abstractions mainly based on goals, softgoals, beliefs and tasks to facilitate the intentional modeling by improving – among other contributions – the goal formation, the like-me recognition and the human practical reasoning specification. Therefore, the ubiquitous application under analysis is investigated by identifying its requirements (i.e. ubiquitous requirements and application-specific requirements). In order to facilitate this elicitation process as well as the requirements' intentional modeling, we also provide the association between some ubiquitous abstractions and the i* abstractions as presented in Chapter 4 – Section 4.1. Figure 5.2 shows the reuse of the *Intentional Modeling Building Block* in a ubiquitous application from the TROPOS'

Early Requirements discipline – *i* Early Requirements Model(s)* – to the TROPOS' Detailed Design discipline – *Detailed Design Requirements Model(s)* based on models evolution.

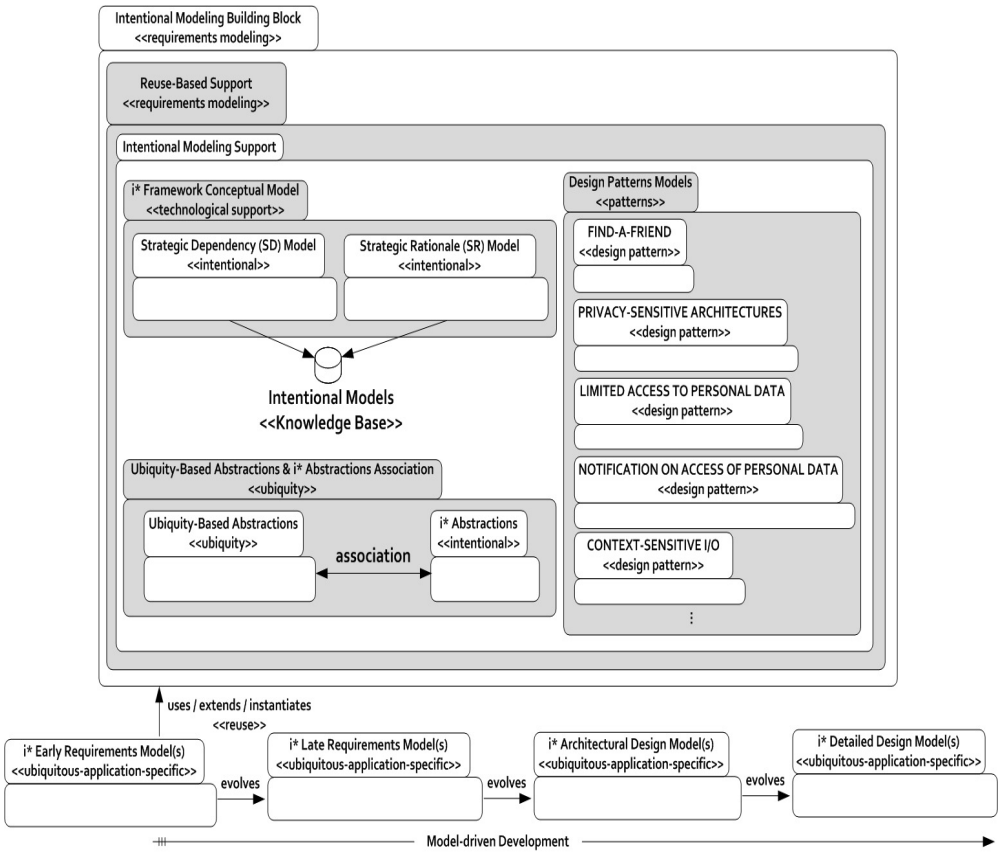

Figure 5.2 - Reuse of the *Intentional Modeling Building Block* in the *Ubiquitous Application Engineering*

5.1.2. NFR Catalogue Building Block Reuse

In order to provide resources for non-functional requirements elicitation, analysis and operationalization, we provide the reuse-based support centered on the NFR Catalogue (Figure 5.3). Based on this catalogue, a relevant REUSE-ORIENTED BUILDING BLOCK is:

- the NFR *Catalogue Building Block* that provides SIGs for non-functional ubiquitous requirements, which can be refined based on the ubiquitous application's investigation. This support promotes and facilitates the model reuse, by also improving the elicitation of non-functional ubiquitous requirements; the determination of their interdependencies with other

specified non-functional ubiquitous requirements; and the knowledge of how to operationalize them with traditional and emergent technologies (e.g. BDI model and MAS).

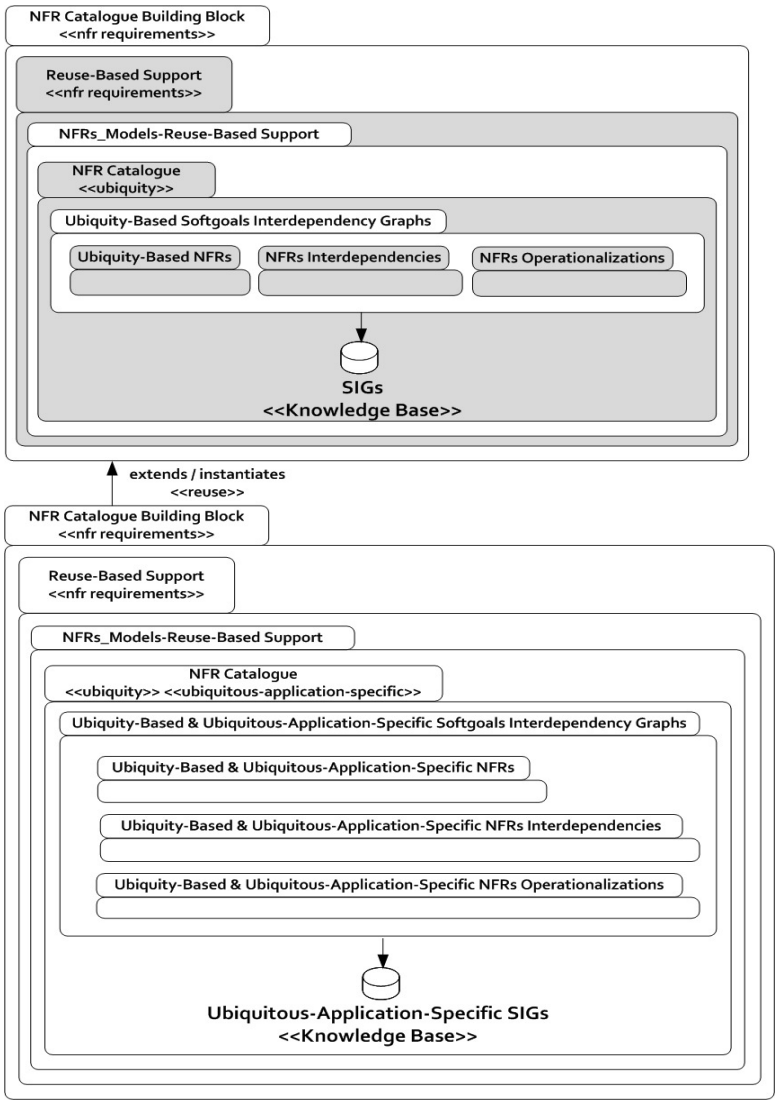

Figure 5.3 - Reuse of the *NFR Catalogue Building Block* in the *Ubiquitous Application Engineering*

5.1.3. Integration Building Block Reuse

We also have the *Integration Building Block* centered on the JADE-LEAP Platform resources (e.g. *Platform Integration Support*). In this field, some support sets (Figure 5.4) that compose this REUSE-ORIENTED BUILDING BLOCK are:

- the *MIDP Device Integration Support* to configure and deal with the integration of MIDP devices by considering the ubiquitous application

under analysis. It deals with the invisibility principle and privacy issues (e.g. accountability to know who is manipulating the user's data) by using the *Split Execution Mode*. In this mode there is a light behavioral agent – i.e. *JADE-LEAP Agent* – inside the device to facilitate the integration process and to better deal with the accountability issue by, for example, using the *DF Service*. This light agent avoids problems with the MIDP device's limited capacity as well as the *DF Service* registers the agent that manipulates the user's data with a unique identifier at the Yellow Pages. It facilitates the agent's identification and traceability at runtime; and

- the *PJava Device Integration Support* to configure and deal with the integration of PJava devices by considering the ubiquitous application. This integration support also respects the invisibility and accountability by using the *Standalone Execution Mode*.

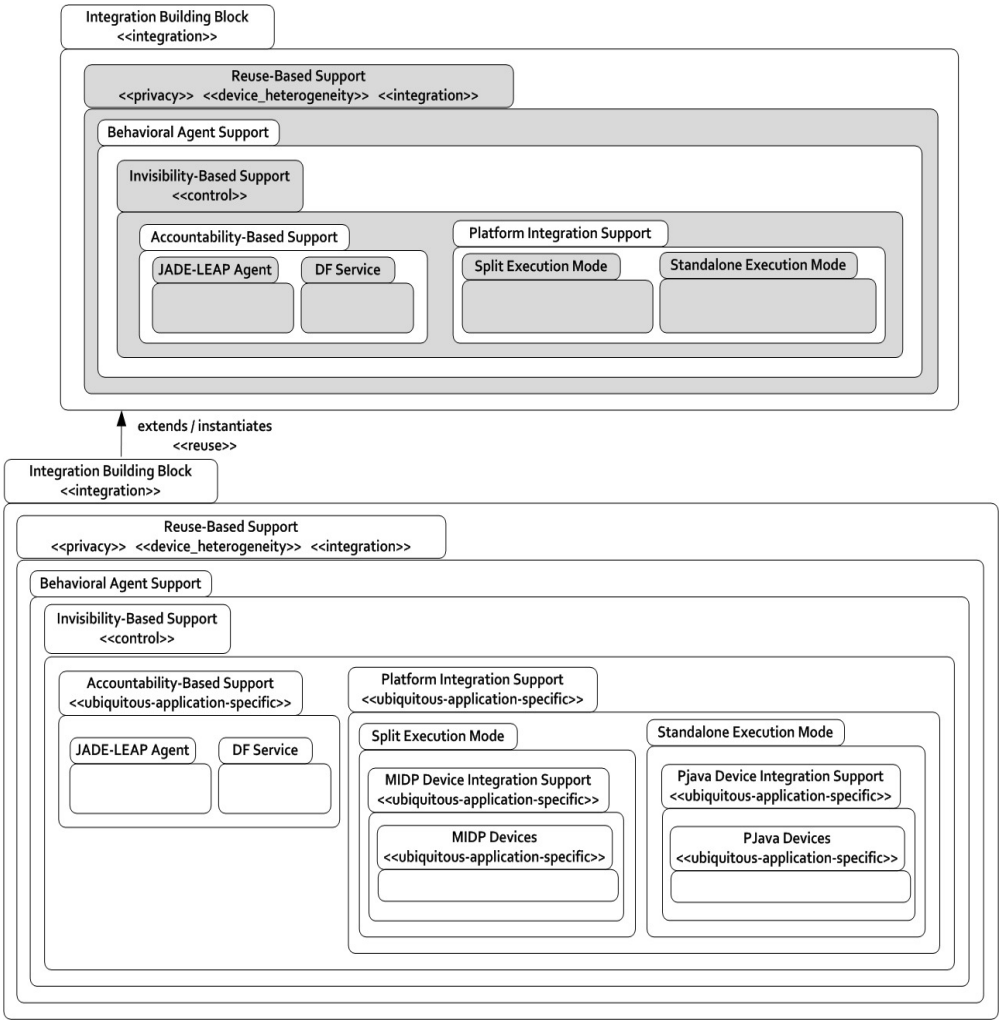

Figure 5.4 - Reuse of the *Integration Building Block* in the *Ubiquitous Application Engineering*

5.1.4.
Intentional Agents' Reasoning Building Block Reuse

Other examples of support sets that also comprise the REUSE-ORIENTED BUILDING BLOCKS (Figure 5.5), but now mainly centered on the JADEX Framework, are presented as follows:

- the *Autonomous Entity Support.* The agents' infra-structure offered by the autonomous entity-support intends to improve the cognitive capacity of the application's agents by using a BDI-based engine; their interoperability and communication by using ontologies; and, consequently, how they deal with the non-functional requirements at runtime and how they balance the invisibility and transparency and personalization and privacy issues; and
- the *Ubiquitous-Application-Based Capability*, for example, the *Privacy-Aware Capability* conceptual model. It suggests the use of capabilities to improve the agents' cognitive capacity in dealing with different users, service providers and their specific business rules. This support – applied to the ubiquitous application and among other contributions – protects the users' personal information related to the ubiquitous application by respecting the users' preferences (e.g. *I would like to share my personal information or I am not comfortable in sharing my personal information with the ubiquitous application*). These preferences can be previously stored into the dynamic database as mandatory, private, or public. Moreover, they can be dynamically investigated, by consulting the user "on the fly." Furthermore, we also have the *DF Capability* to register and deregister the agents at the platform with a unique identifier, which can be used as an accountability resource to attribute responsibilities to the agents; the *Mobility Capability* to deal with the intrinsic mobility of the users by providing location-aware resources to improve the invisibility into the ubiquitous application; the *AMS Capability* to automate the creation of the agents at runtime – invisibility-centered resource – by maintaining control of the agents' life-cycle – dependability- & security-centered resource. It is important to contextualize that the traceability of the agents' activities – by manipulating personal information of the users – is really improved by using this latter resource.

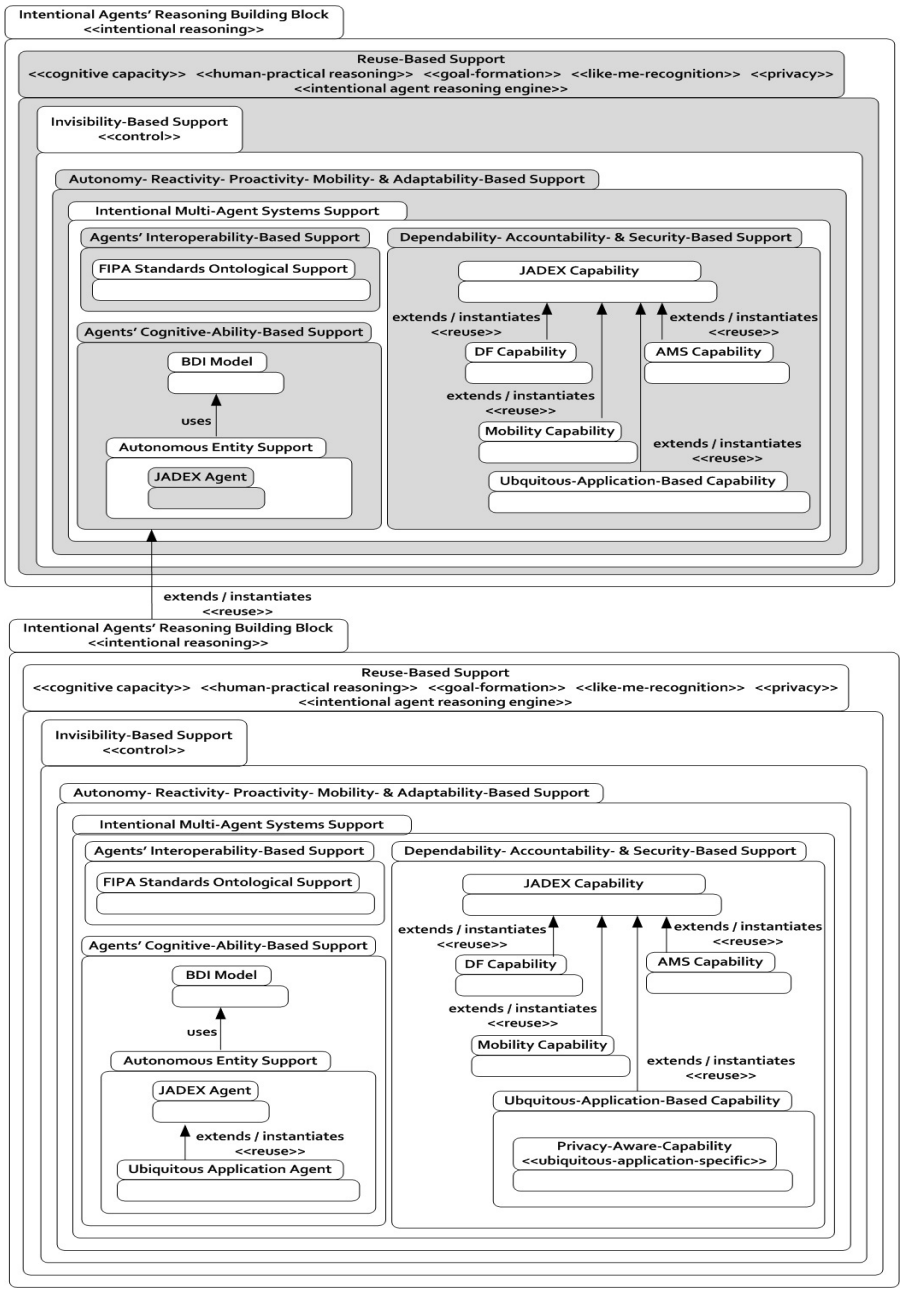

Figure 5.5 - Reuse of the *Intentional Agents' Reasoning Building Block* in the *Ubiquitous Application Engineering*

5.1.5. Fuzzy-Logic-Based Package Reuse

Another example of support set that enhances the REUSE-ORIENTED BUILDING BLOCKS is centered on a Fuzzy-Logic Library (Figure 5.6). As we concentrate our efforts on the usage of intentional agents based on the **B**elief-**D**esire-**I**ntention (BDI) model, our approach already contemplates a certain degree

of cognition. Therefore, it supports the goal formation (Dignum and Conte 1997) and the like-me recognition (Gordon 2005) driven by the user's beliefs, desires and intentions. However, there are interesting ways to improve the human-practical reasoning (Bratman 1999). In order to deal with this field, we also developed a reuse-based support – *Fuzzy_Logic-Based Support* package – focused on fuzzy-logic-based reasoning. This support is centered on different non-functional issues (e.g. security and price) and on the usage of a specific fuzzy-logic library (Bigus and Bigus 2001; Serrano and Lucena 2010a), which can be extended or instantiated to better attend the ubiquitous application's issues.

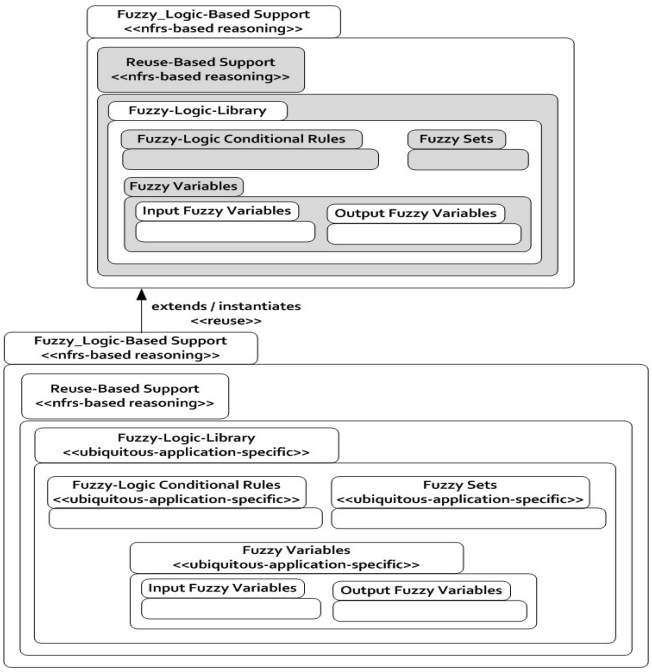

Figure 5.6 - Reuse of the *Fuzzy-Logic-Based* package in the *Ubiquitous Application Engineering*

The fuzzy variables, fuzzy sets and fuzzy rules are specified by considering the ubiquitous application under analysis. According to our experimental research, the application-issues-aware variables, sets and rules comprise a suitable technological support to improve the agents' reasoning at runtime. The intentional agents perform their tasks based on this technological support by dynamically deciding which is the best way to achieve the delegated goal. For each agent's decision, the fuzzy conditional rules are executed at runtime by considering the context under analysis and the pre-defined fuzzy sets and fuzzy variables.

Simplifying the process, we can say that an intentional agent – that represents the user as a personal agent – collaborates with the agents responsible

for content providers (e.g. images, videos and files) or service providers (e.g. download service and payment service) in order to obtain information based on different quality criteria (e.g. security, price and download time issues). The personal agent analyzes the received information centered on its beliefs base, the ubiquitous profiles – that contain the user's preferences (e.g. which quality criterion is more important for her/him) and privacy policies, the device features, the network specification and the contract information between the user and the content provider or the service provider – and by running the fuzzy-logic conditional rules with the acquired information. Therefore, the decision is determined by considering, for example, that the security criterion is more important than the price as well as the price criterion is more important than the download time. Finally, the personal agent establishes contact with the agent responsible for the chosen content provider or service provider to receive the desired content or service. It is also possible to perform adaptations to adequate, for example, the content according to the ubiquitous profiles. Finally, the adapted content is provided to the user from her/his device.

5.1.6. Dynamic Interface Construction Building Block Reuse

An interesting resource of our approach is the *Dynamic Interface Construction Building Block* that also composes the REUSE-ORIENTED BUILDING BLOCKS. It is a reuse-oriented support to deal with the construction of interfaces by taking into consideration the device features and the user preferences. The construction is performed by intentional agents "on the fly" centered on the *FIPA Standards Ontological Support*. Figure 5.7 shows the use of this building block in the *Ubiquitous Application Engineering*. We zoom in the *GUI Generic Ontology* of the *Domain Engineering of Ubiquitous Applications*. In the *Ubiquitous Application Engineering*, it is also possible to just reuse the *GUI Generic Ontology* as well as the *MIDP GUI Ontology* as they are defined. Furthermore, if the application under development demands an ontology with very particular interface elements – that do not match with the pre-defined interface elements of the *GUI Generic Ontology* and/or the *MIDP GUI Ontology*, it is recommended to define this specific ontology – *GUI Ontology* – by constructing its asserted model

(Protegè 2011), then associate this ontology with the *GUI Generic Ontology* by using heuristics. Finally, it is necessary to develop a service to adapt the interface elements of the desire ontology based on the heuristics.

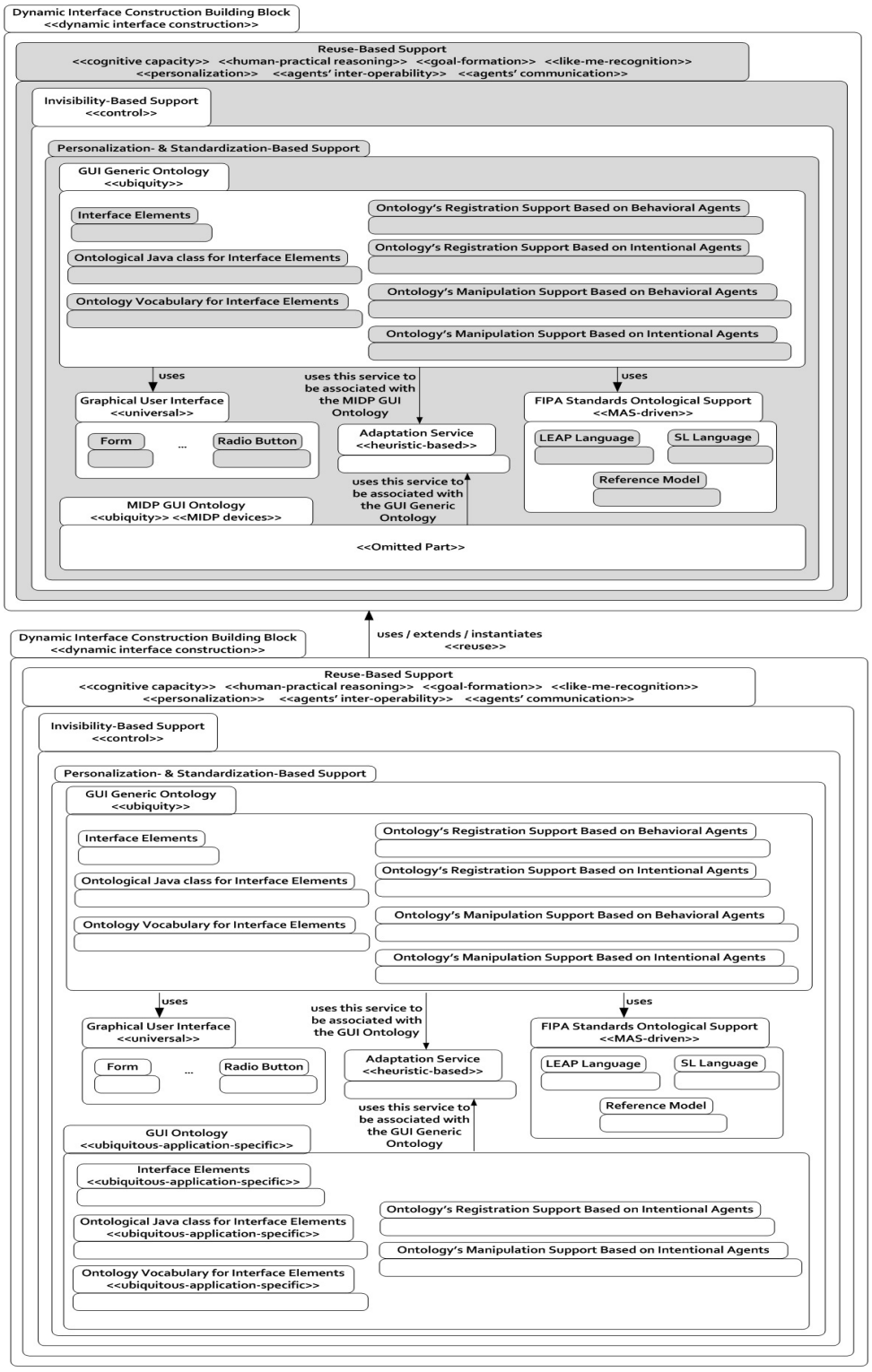

Figure 5.7 - Reuse of the *Dynamic Interface Construction Building Block* in the *Ubiquitous Application Engineering*

5.1.7.
Ubiquity Issues Building Blocks Reuse

According to our proposal, different ubiquity-based frameworks can be reused to deal with specific ubiquitous issues, such as content adaptation and context awareness. This kind of practice can reduce the time spent in the development and the software engineers' effort. As presented on Chapter 4, we provide a *Ubiquity Issues Building Blocks* composed of a reuse-based support centered on *Ubiquity-Based Frameworks* package. It represents another REUSE-ORIENTED BUILDING BLOCK. Figure 5.8 illustrates the reuse of a specific framework – i.e. IFCAUC – to deal with content adaptability in ever-changing contexts. This intentional-MAS-driven framework is offered as an API that can be instantiated or extended to better attend the ubiquitous application under analysis.

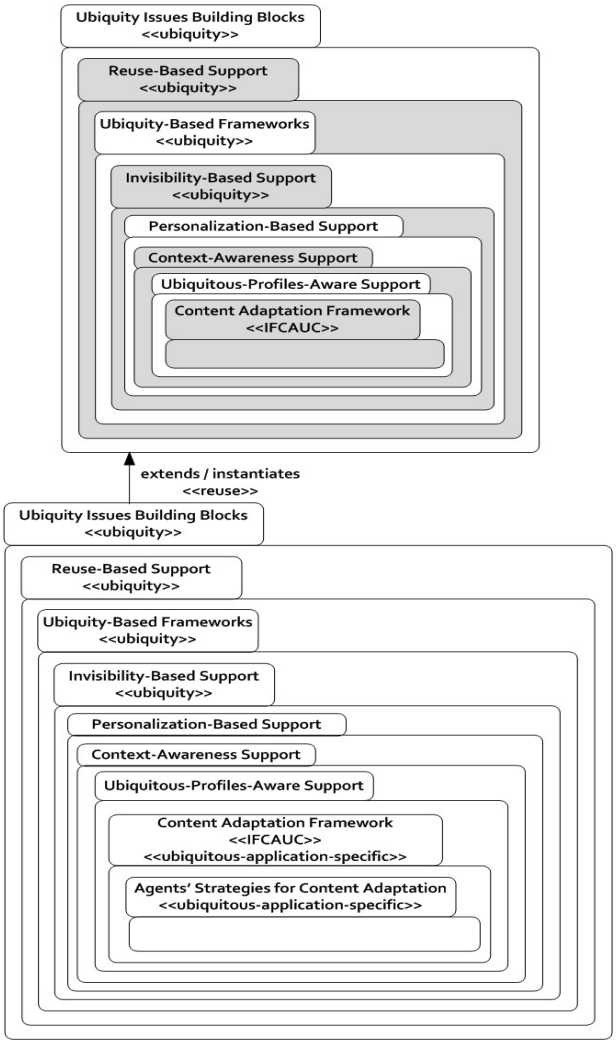

Figure 5.8 - Reuse of the *Ubiquity Issues Building Block* in the *Ubiquitous Application Engineering*

Focusing our attention on plans deliberation, goals, beliefs and events, the BDI model – used as the main support in the IFCAUC – is viewed as appropriate to implement a complex ubiquitous application based on the MAS and Goal-Orientation paradigms, since it represents the agents' society as a natural metaphor for human reasoning. This natural association inspired us to develop a context-aware infrastructure to improve the content adaptation in ubiquitous scenarios. According to the ubiquitous application's issues, the IFCAUC can be extended or instantiated by, for example, allowing intentional agents to perform adaptations based on different strategies (e.g. resizing and transcoding). The IFCAUC agents make their decisions mainly centered on the user profile and the device profile. Figure 5.9 illustrates both agents' adaptation strategies using a pyramid representation and by considering that the desired ubiquitous application must provide support to audio, image, text and video contents. Moreover, the application under analysis demands efforts to deal with heterogeneous devices (e.g. MIDP devices, PJava devices and Jse devices).

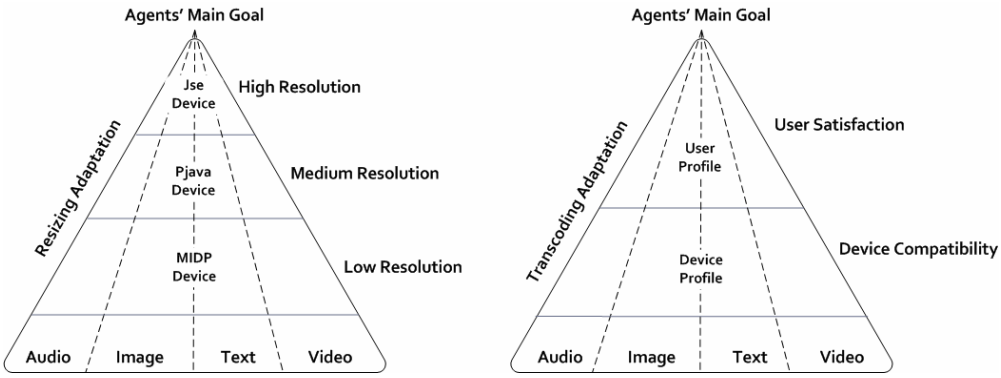

Figure 5.9 - Agents' adaptation strategies

In adaptation based on resizing, the agents' main goal is adapt the content according to the device screen resolution. MIDP Device, for example, normally has low memory and processing capacity. Therefore, intentional agents must adapt the contents in MIDP devices by considering the low resolution need.

In adaptation based on transcoding, the agents' main goal is to satisfy the user. However, if it is not possible because of the device features, then the agents try to achieve their goal by being compatible with the device. Therefore, the agents adapt the content by converting its type according to: first, the user profile information (e.g. type desired by the user); and/or alternatively, the device profile information (e.g. type accepted by the device).

5.1.8.
Dynamic Database Building Block Reuse

The *Dynamic Database Building Block,* which enriches the REUSE-ORIENTED BUILDING BLOCKS in the persistence layer, can be reused by mainly extending or instantiating the *Dynamic Data Model package* based on the ubiquitous application's issues as presented in Figure 5.10.

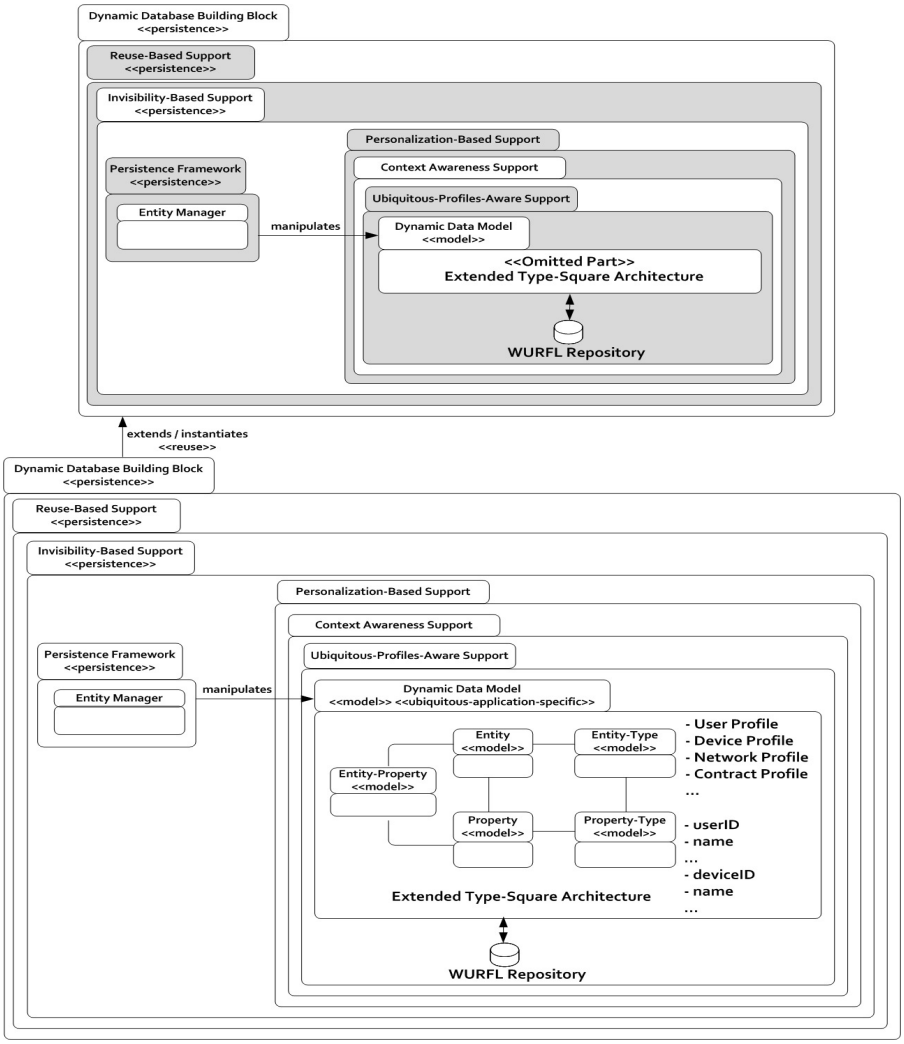

Figure 5.10 - Reuse of the *Dynamic Database Building Block* in the *Ubiquitous Application Engineering*

In order to illustrate the ubiquitous profiles based on the dynamic data model, we have the *User Profile, Device Profile*, *Network Profile* and *Contract Profile* as Entity-Type(s) (Figure 5.11). Moreover, for each Entity-Type, we can specify its attributes. For example, the *userID*, *name*, *login*, *password*, *address* and other attributes as the Property-Type(s) of the *User Profile*. The Entity and Property can be instantiated centered on the ubiquitous application under analysis.

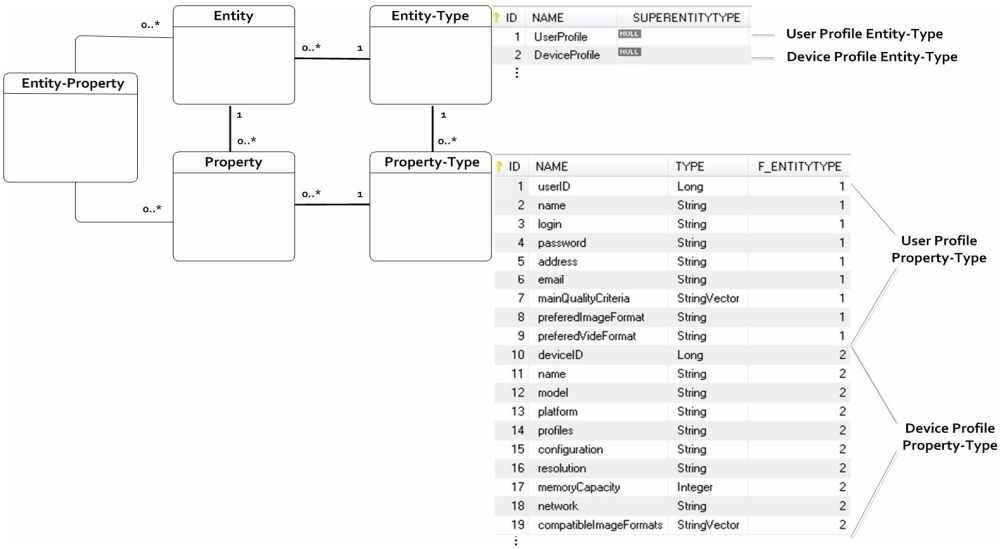

Figure 5.11 - *Ubiquitous Profiles* and the *Dynamic Data Model*

More details of the REUSE-ORIENTED BUILDING BLOCKS are provided in Chapter 6, in which they are used to develop a dental clinic ubiquitous application. Furthermore, these artifacts are reused according to the architecture presented as follows.

5.2. Reuse-Oriented Architecture

Summarizing our proposal centered on the development with reuse, we can consider a reuse-oriented architecture that illustrates the usage or extension or instantiation of the described building blocks in the *Ubiquitous Application Engineering* (Figure 5.12). The architecture is based on the **M**odel-**V**iew-**C**ontroller (MVC) architectural pattern, which was firstly proposed in 1978/79 by Trygve Reenskaug at XEROX PARC. It allowed us to separate the business logic and the view logic. In the View, we concentrate the interface-based artifacts (e.g. *Dynamic Interface Construction Building Block*, *Heterogeneous Devices* and their *Users*). In the Controller, we have the building blocks that are the brain of our intentional-MAS-driven approach, such as the *Integration Building Block* and the *Intentional Agents' Reasoning Building Block*. In the Model, we represent the building blocks as well as some building blocks packages with specific business rules, such as the *Dynamic Database Building Block*, the *Ubiquity Issues Building Blocks*, the *Intentional Modeling Building Block*, the *BDI Model*, and the *JADEX Capability*. Moreover, stereotypes are used to facilitate the representation of the

building blocks obtained from the *Domain Engineering of Ubiquitous Applications* (*ubiquity* stereotype) and their specialization in the *Ubiquitous Application Engineering* (*ubiquitous application* stereotype).

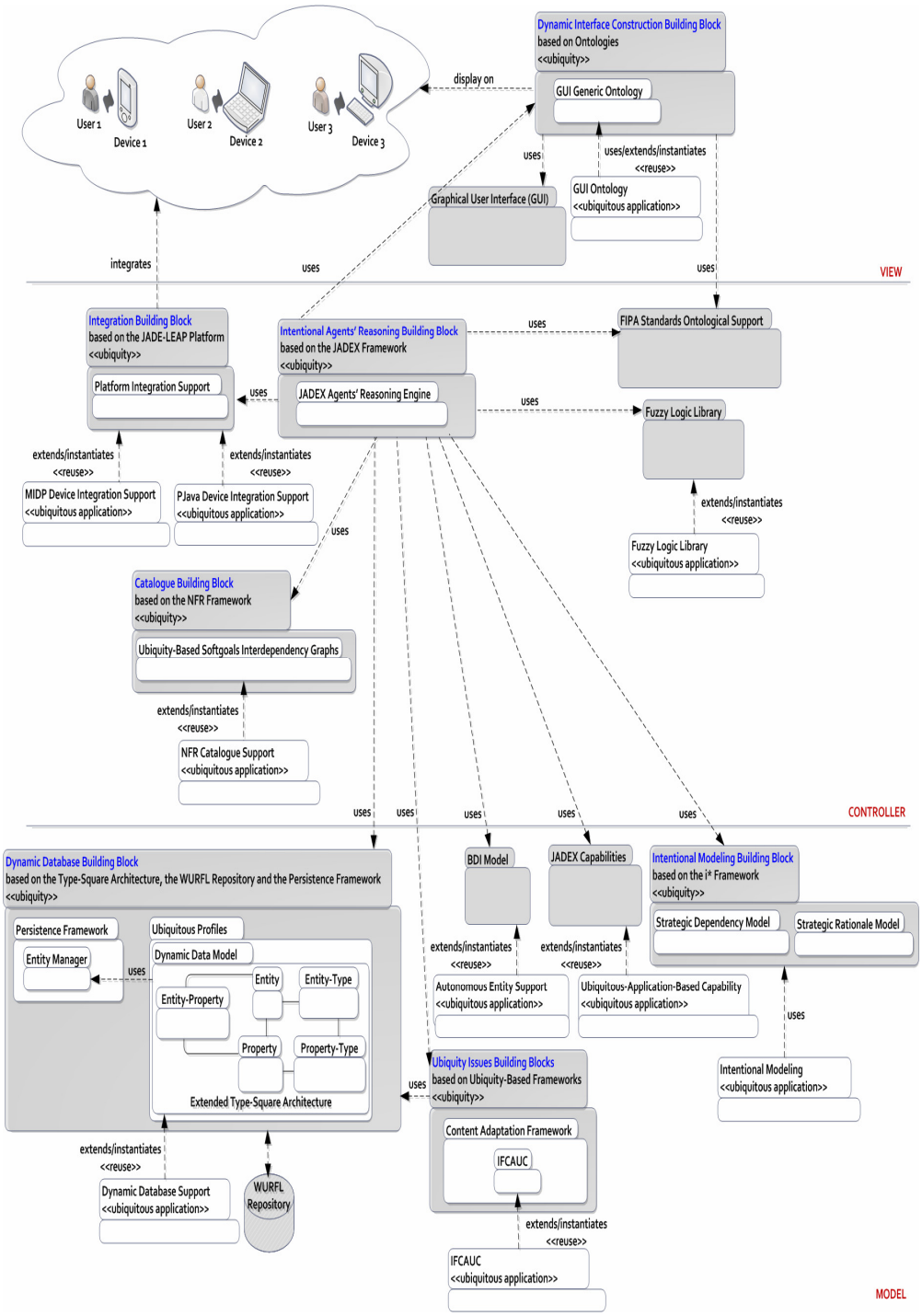

Figure 5.12 - The proposed *Reuse-Oriented Architecture*

Only to exemplify the reuse-oriented architectural representation, we can take a look on the proposed *Dynamic Interface Construction Building Block*

support. We zoom in its *GUI Generic Ontology* package reused in the *Ubiquitous Application Engineering* by generating/developing a specific *GUI Ontology* based on the ubiquitous application under analysis. This package helps the software engineers to dynamically construct interfaces by also adapting them for heterogeneous devices according to the ubiquitous profiles and other context-aware information. Then, the adapted interfaces are displayed on the devices' screen anywhere and at any time they were necessary.

5.3. Incremental and Systematic Development Life-Cycle

The development of the ubiquitous applications is conducted by following an incremental and systematic life-cycle centered on the disciplines of the TROPOS Model-Driven Development (Bertolini et al. 2006). However, we incorporated some Agile Methods principles (e.g. iterative and incremental model) and some adjustments to deal with the systematic development of ubiquitous applications based on reuse in different abstraction levels (e.g. model reuse in higher abstraction levels and framework reuse in lower abstraction levels). Therefore, we divided our incremental and systematic development approach in phases with those disciplines.

In the *Requirements Phase*, the software engineers perform the Early & Late Requirements disciplines. In the *Design Phase*, the software engineers perform the Architectural & Detailed Design disciplines. In the *Code Phase*, the software engineer concerns with the Implementation discipline. Thus, the Test discipline is performed in the *Evaluation Phase* with the stakeholders' interaction to verify and validate the ubiquitous application or part of it if the application is in intermediary stage of development. Furthermore, the application will be deployed and evolved to follow technological trends. The phases are not obligatorily performed in a sequential order. They can be (and we recommend it) performed in parallel as expected by an incremental model, presented in Figure 5.13.

Thanks to the incremental nature of our life-cycle, we consider a Δ (*delta*) for each phase in each iteration. Therefore, we have the *Requirements Phase* (Δ_R), the *Design Phase* (Δ_D), the *Code Phase* (Δ_I) and the *Evaluation Phase* (Δ_T). The Δ represents the development complementary part to conclude the phase under

analysis. For example, the Δ_R represents the development complementary part to conclude the *Requirements Phase* composed of the Early & Late Requirements disciplines.

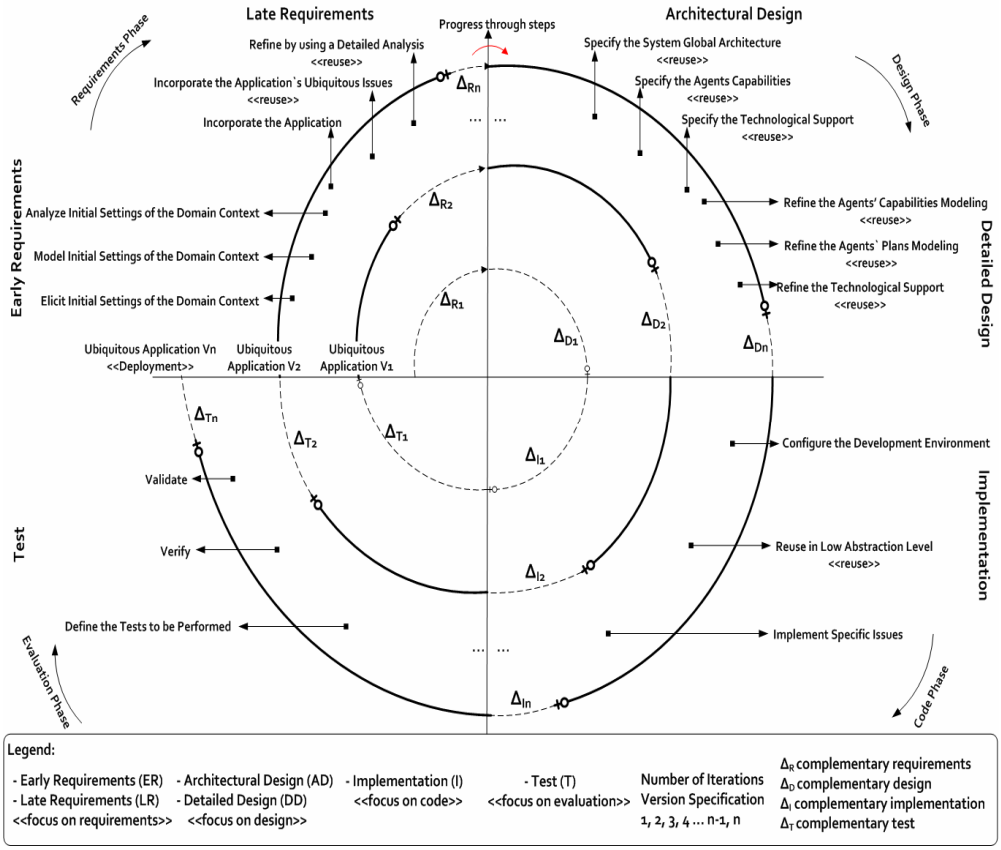

Figure 5.13 - Life-cycle: *Incremental and Systematic Development* (adapted from (Boehm, 1986))

As follows we briefly describe each discipline of the life-cycle. It is important to consider that our development approach focuses their main contributions on the disciplines underlined. Therefore, we provide additional details of them.

- Early Requirements (ER) – Figure 5.14

  This discipline is centered on TROPOS and comprehends the initial organization setting of the cognitive domain under analysis. Therefore, it is necessary to perform this discipline before the introduction of the application-to-be in the goal-oriented analysis process. The cognitive domain organization setting is elicited by using different elicitation techniques (e.g. brainstorming, open/close questionnaires and observation). Moreover it is modeled and analyzed by using the i* and the NFR Frameworks. The result is the Early Requirements Models.

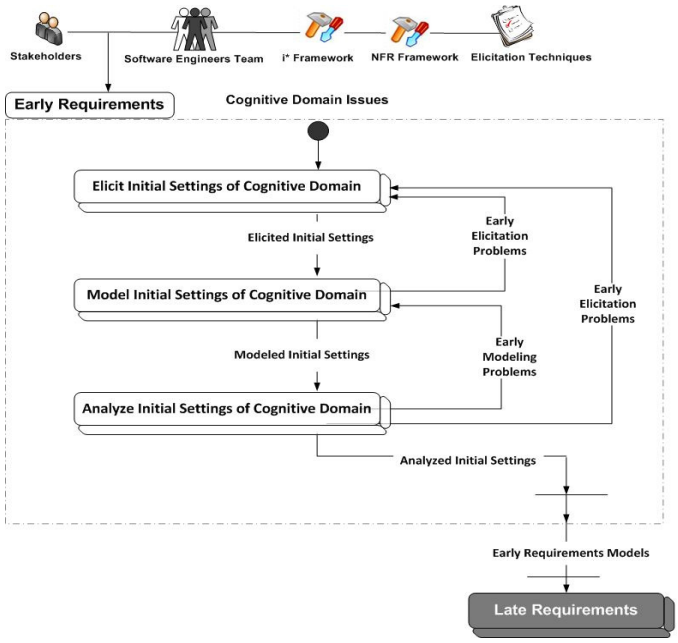

Figure 5.14 - *Early Requirements* discipline

- Late Requirements (LE) – Figure 5.15

  This discipline is also centered on the TROPOS. Here, the application-to-be is introduced in the domain model. Therefore, it is necessary to specify the dependencies among the application and other modeled details. These dependencies represent the "obligations" of the application towards its environment and actors. In this field, it is recommended the use of the i* Framework to model the dependencies, the use of the NFR Framework to model the non-functional requirements and the use of the scenarios representation and C&L Tool[1] to compose an adequate view of the application's main activities. Thus, the focus is on the application within its operating environment. Moreover, we propose the introduction of the application's ubiquitous issues. At this moment, the ubiquitous issues are specified and modeled. Furthermore, an internal analysis of this ubiquitous application is performed by using goal analysis techniques. From the elicitation to the analysis of the ubiquitous issues, our approach suggests the reuse of the support provided by the *Intentional Building Block* and the *NFR Catalogue Building Block*. Finally, it is important to perform some refinements, whose necessity was identified during the detailed analysis

[1] **C**enários & **L**éxico (C&L) is an editor of scenarios and lexicon (http://pes.inf.puc-rio.br/cel/)

activity. The software engineers, for example, can choose SIGs from the NFR Catalogue with non-functional requirements that match with the application's ubiquitous issues and then perform a detailed analysis to solve conflicts with the stakeholders' participation. Based on this process, the ubiquitous application's requirements are incrementally specified in each iteration to obtain the Late Requirements Models.

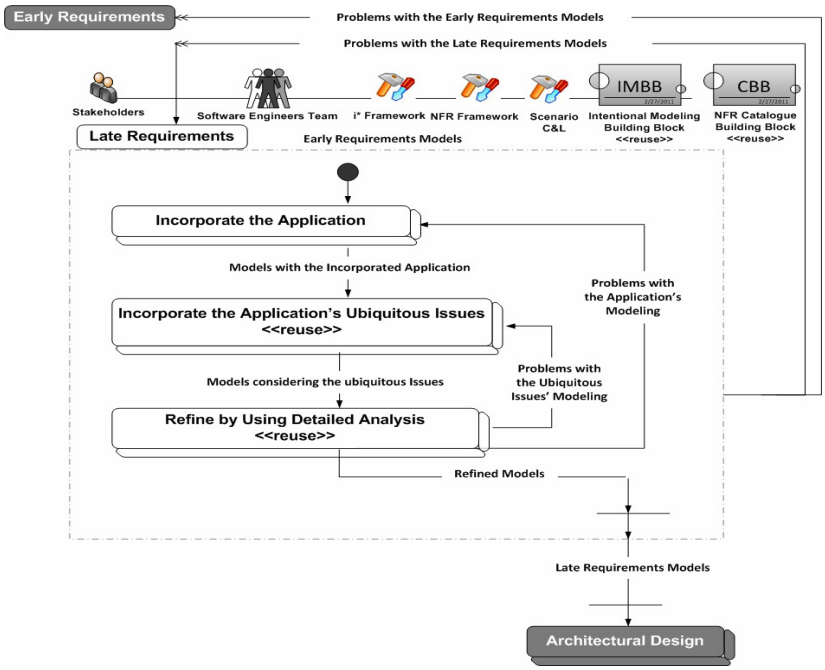

Figure 5.15 - *Late Requirements* discipline

- Architectural Design (AD) – Figure 5.16

  This discipline is centered on TROPOS. Now, it is important to specify the global architecture of the ubiquitous applications. Thus, sub-applications of the application-to-be and their relationships with external sub-applications and actors must be specified. Moreover, the set of capabilities of the application are represented by both means-ends goals/tasks relationship and task/softgoal contributions. Here, it is also recommended the use of the *Intentional Modeling Building Block* and the *NFR Catalogue Building Block* in order to improve the i* models and the SIGs obtained from the Early & Late Requirements disciplines. The *Catalogue Usage Method* (presented on Section 4.2) can facilitate this process by providing guidelines to explore, collect, model, operationalize and validate the NFR Catalogue's SIGs. Finally, it is recommended to specify the main technologies (e.g. intentional Multi-Agent Systems and BDI model) that

will be used throughout the application's development. The operationalizations provided by the NFR Catalogue's SIGs can contribute to the technological support determination. It is not necessary to specify the technological support for the ubiquitous concerns, as the software engineer will have the opportunity to do that in the next discipline, *Detailed Design*. The process's result is the Architectural Design Models.

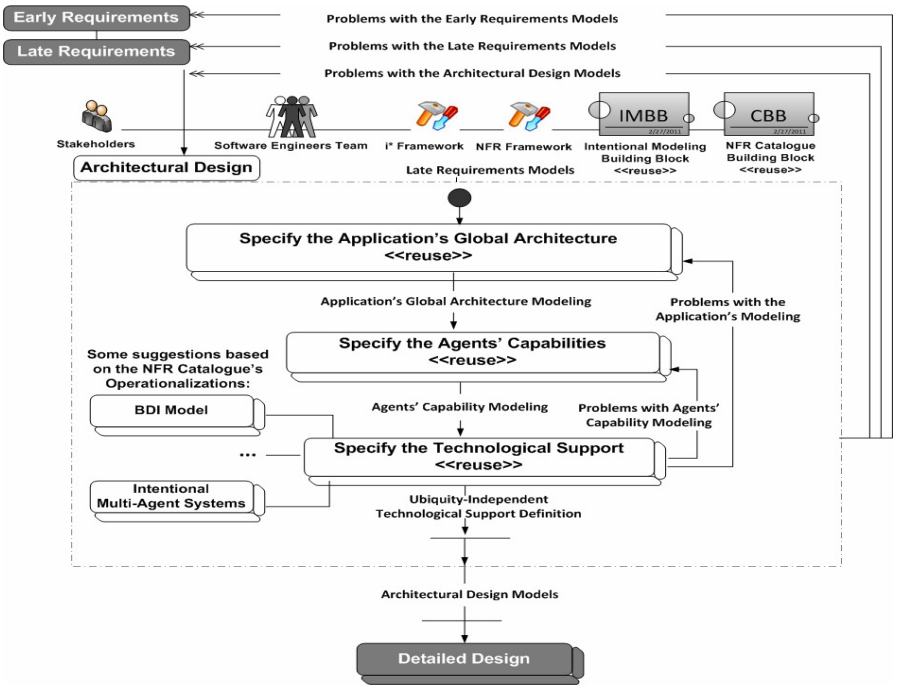

Figure 5.16 - *Architectural Design* discipline

- Detailed Design (DD) – Figure 5.17

  This discipline is centered on TROPOS. Here, AND-OR decomposition diagrams are refined in order to complement the specification of the agents' capabilities. In other words, this discipline deals with the specification of the agents' micro level by refining their rationale. We also propose that the software engineers define the technological support to deal with ubiquitous issues. In this field, the *Integration Building Block* based on the JADE-LEAP Platform, the *Intentional Agents' Reasoning Building Block* based on the JADEX Framework and a Fuzzy Logic Library, and the *Ubiquity Issues Building Blocks* based on Ubiquity-based Frameworks (e.g. IFCAUC) are some examples of reuse-driven support sets. They provide – among other contributions – conceptual models that can be reused by facilitating the refinement process (i.e. the specification

of the agents' micro level). Moreover, the operationalizations specified in the NFR Catalogue's SIGs as well as the intentional design patterns provided by respectively the *NFR Catalogue Building Block* and the *Intentional Modeling Building Block* can contribute to the technological support determination for ubiquitous issues. The result of this process is the Detailed Design Models.

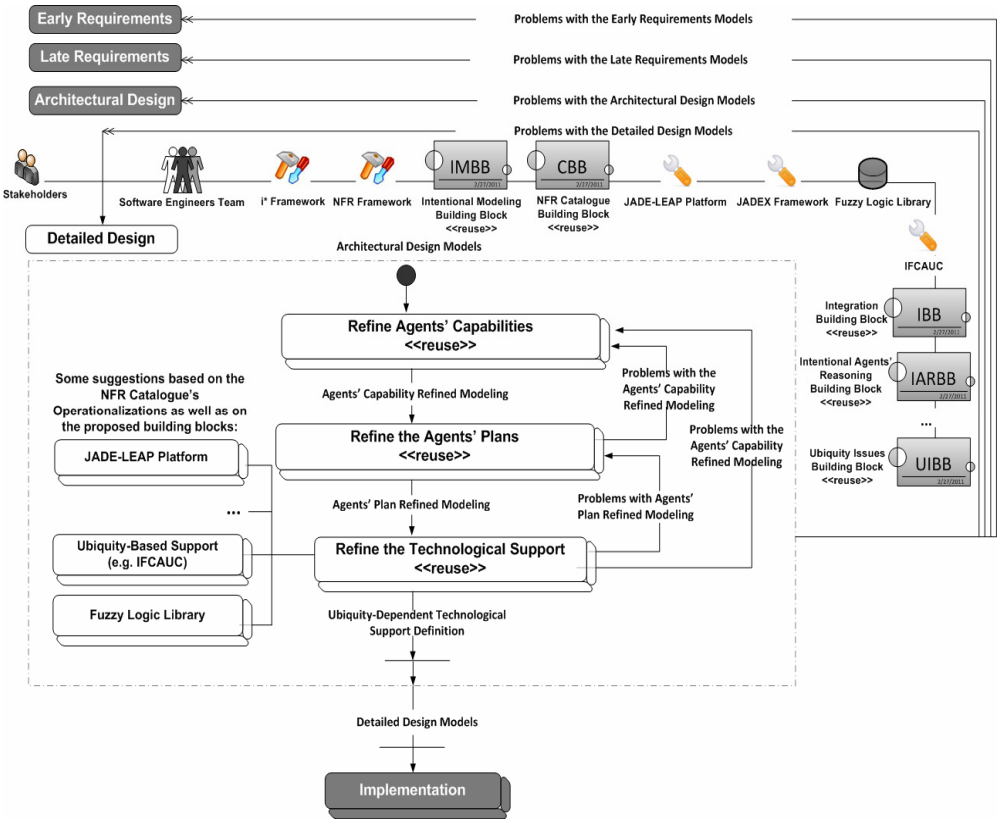

Figure 5.17 - *Detailed Design* discipline

- Implementation (I) – Figure 5.18

  This discipline is centered on TROPOS. It concerns with the implementation of the ubiquitous application. In order to facilitate this process, we offer the association between the higher abstraction levels (i.e. modeling) and the lower abstraction levels (i.e. code) in our *Intentional Modeling Building Block*. In Chapter 4 – Section 4.1, we already explained the association between some abstractions of intentional ubiquitous applications and the i* abstractions. This association is a punctual contribution of our proposal. However, the association between the i* abstractions and the JADEX BDI model abstractions – presented in Figure 5.19 – is based on (Serrano et al. 2011b). They offer heuristics centered on

this association to facilitate the implementation of intentional applications from their i* modeling. Summarizing the activities of this discipline, we can focus on: (i) the development environment's configuration; (ii) the computational support reuse in low abstraction level – e.g. *Intentional Agents' Reasoning Building Block* based on the JADEX Framework and the *Fuzzy-Logic-Based Support*, *Dynamic Interface Construction Building Block* based on Ontologies and *Dynamic Database Building Block* based on the Type-Square Architecture, the WURFL Repository and a Persistence Framework – to avoid work replication in the implementation of commonly found issues; and (iii) the specific issues implementation – the application-to-be probably have some particular issues that require special attention. In these specific situations it can be difficult to find ready and extraordinary solutions. Therefore, the software engineers must dedicate their time to deal with those issues. The process's result is the code of the intentional-MAS-driven ubiquitous application.

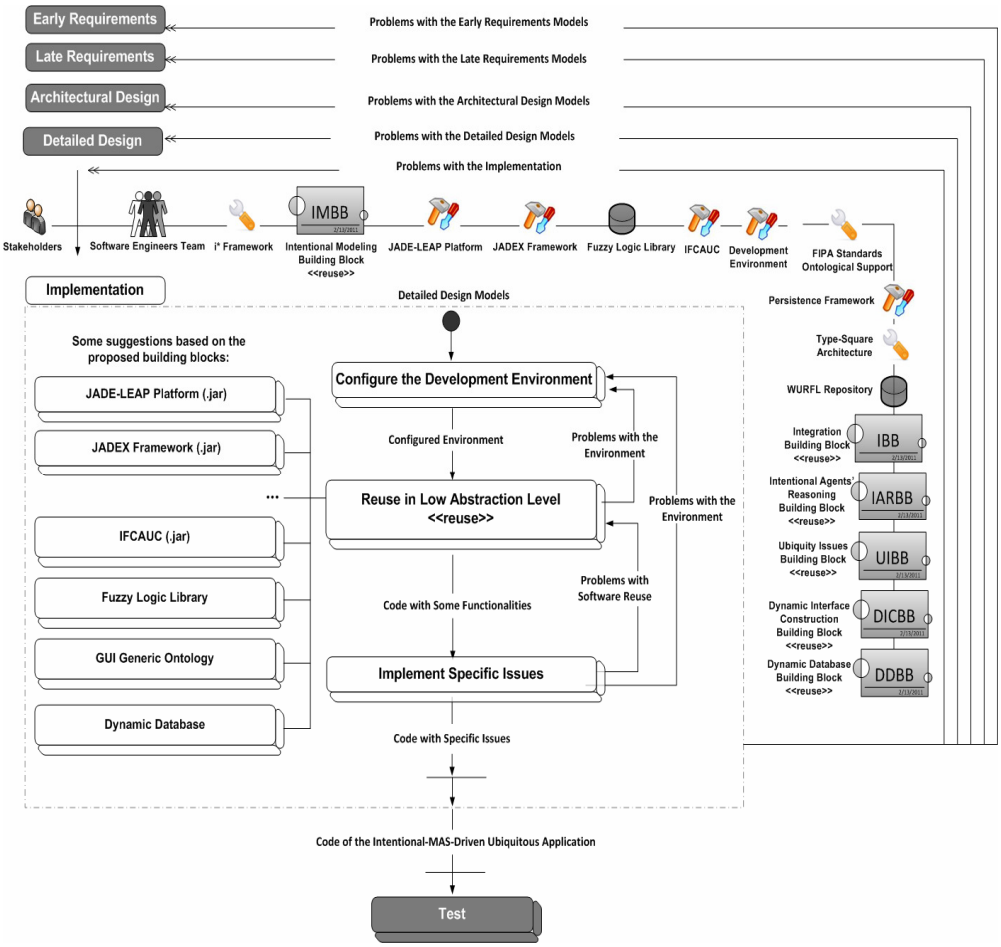

Figure 5.18 - *Implementation* discipline

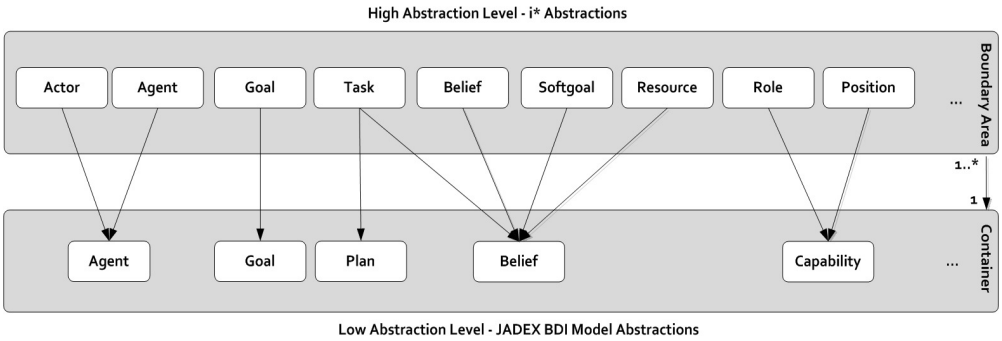

Figure 5.19 - Association between i* abstractions and *JADEX BDI Model* abstractions

- Test (T) – Figure 5.20

  This discipline concerns with the topics: (i) find and document failures to avoid their replication; (ii) evaluate the assumptions made in the previous design disciplines – i.e. if the ubiquitous application works as designed, using interactions with the users, their satisfaction and other non-functional requirements as parameters; and (iii) evaluate if the requirements are correctly implemented. Here, it is important to keep in mind some questions, such as: (1) *How could the ubiquitous application break?* Performing tests (e.g. stress test) are appropriate to evaluate this kind of efforts; (2) *Are there some situations in which the users' satisfaction degree is low?* Tests with the users' participation can be applied to this field; (3) *Are there specific scenarios in which the ubiquitous application could fail?* Investigate the ubiquitous application behavior on real environments or simulated ones can be an appropriate way to analyze when it fails. The *Test* discipline focuses on the definition of the tests to be performed; the verification of the ubiquitous application – e.g. identification of incoherent specifications by re-analyzing the application's modeling using the i* Framework and its goals analysis technique as well as the NFR Framework and its propagation rules – presented in the last activity of the *Catalogue Usage Method* (Section 4.2); and the validation of the ubiquitous application – e.g. identification of inconsistencies in the ubiquitous application by using specific tests and tools (e.g. JADEX Control Center[2]). The process's result is the ubiquitous application version "i," obtained from "i" iterations of the life-cycle.

[2] **J**adex **C**ontrol **C**enter (JCC) represents the main access point for all available Jadex runtime tools. It provides several options to evaluate Java expressions contained in Agent Definition Files.

- Deployment (D)

  This discipline concerns with the implantation of the developed ubiquitous application. The proposed thesis is not focused on this discipline. We only represent it into the life-cycle in order to present a complete overview of the development process – i.e. from the application's investigation to the application's implantation.

- Evolution (E)

  This discipline concerns with the evolution of the ubiquitous application – after its deployment. The evolutionary maintainability is particularly important in applications for ever-changing contexts, such as mobile, pervasive and ubiquitous context. Again, the proposed thesis is not focused on this discipline. We only represent it into the life-cycle in order to present a complete overview of the development process – i.e. from the application's investigation to the application's evolution.

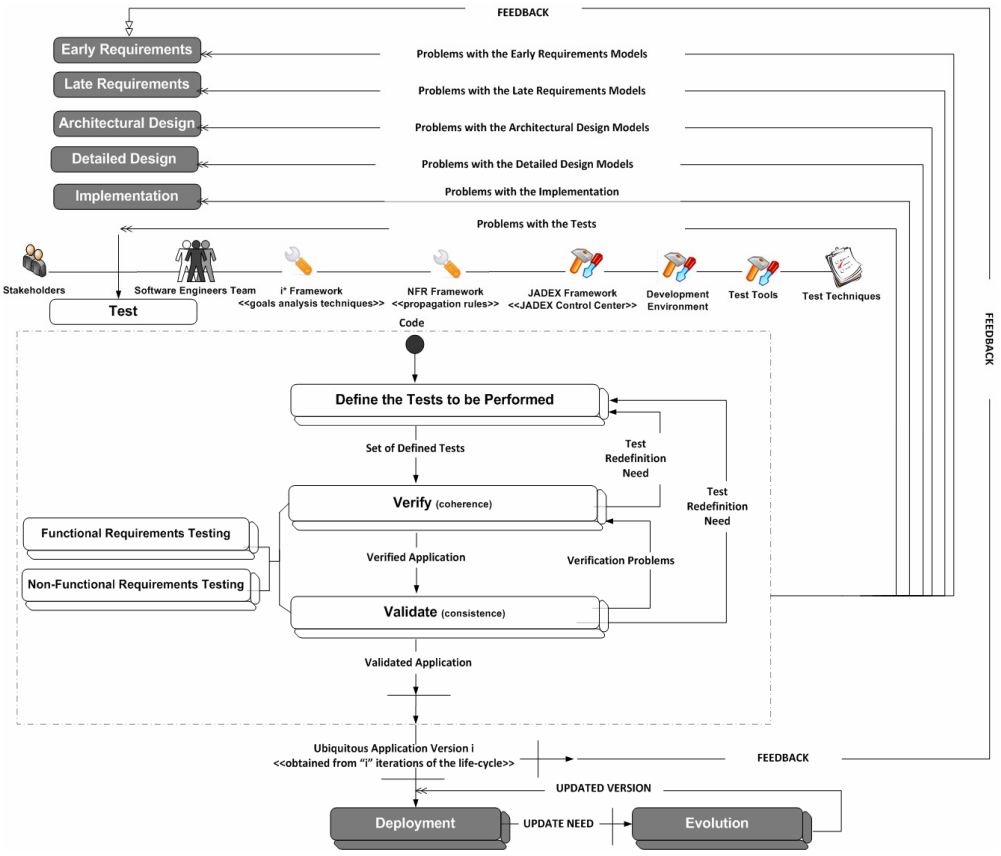

Figure 5.20 - *Test* discipline

Our idea with the proposed life-cycle model is to deal with the necessity of immediate results in ubiquitous contexts. The users' interests in ubiquitous

contexts as well as their devices, the network support, the services or contents and other elements are in constant evolution. This intrinsic ever-changing feature of ubiquitous contexts demands a development approach that quickly presents some results, which can be evaluated by the users and, if necessary, refined. It previously alerts about possible errors by avoiding their replications. Moreover, it helps on the identification of good practices by allowing the reuse of them. Considering a metaphor, in which a ubiquitous application is a house. The proposal is not to construct the house (or the ubiquitous application) as a whole and then show it to the client to know if everything is according to her/his expectations. Instead, we suggest an incremental construction, in which a room is constructed by firstly raising a wall. Then it is presented to the client. The feedback of the client can imply on: (i) discarding undesired practices; (ii) refining the wall's construction; and (iii) reusing good practices.

5.4. Closing Remarks

This Chapter presents our incremental and systematic development for intentional ubiquitous applications. We zoom in the architecture and the life-cycle of the development. The architecture is illustrated by using the MVC architectural pattern and focusing on the reuse of building blocks, obtained from the *Domain Engineering of the Ubiquitous Applications*. The life-cycle is represented as a spiral model by emphasizing the incremental development of ubiquitous applications – *Ubiquitous Application Engineering* – based on TROPOS's disciplines, which are performed by following a model-driven iterative process.

The *Ubiquitous Application Engineering* deals with the incremental and systematic development with reuse. Therefore, our reuse-oriented approach suggests the construction of intentional ubiquitous applications centered on the offered building blocks. These building blocks are mainly composed of conceptual models used as controls in the *Ubiquitous Application Engineering*; and frameworks, libraries, catalogues and patterns used as mechanisms in the *Ubiquitous Application Engineering*. In the next Chapter, we develop a dental clinic ubiquitous application based on the proposed reuse-oriented approach for incremental and systematic development of intentional ubiquitous applications.