Intelligent Systems: From Theory to Practice
Contents

Tagging and Fuzzy Sets ....................................................... 1
Ronald R. Yager, Marek Reformat

Intelligent Control of Uncertain Complex Systems by
Adaptation of Fuzzy Ontologies ........................................... 19
Mincho Hadjiski, Vassil Sgurev, Venelina Boishina

NEtwork Digest Analysis Driven by Association Rule
Discoverers ................................................................. 41
Daniele Apiletti, Tania Cerquitelli, Vincenzo D’Elia

Group Classification of Objects with Qualitative Attributes:
Multiset Approach ....................................................... 73
Alexey B. Petrovsky

Empirical Evaluation of Selected Algorithms for
Complexity-Based Classification of Software Modules and
a New Model .......................................................... 99
Jian Han Wang, Nizar Bouguila, Tuoufik Bdiri

A System Approach to Agent Negotiation and Learning ...... 133
František Čapkovič, Vladimir Jotsov

An Application of Mean Shift and Adaptive Control to
Active Face Tracking .................................................. 161
Ognian Boumbarov, Plamen Petrov, Krasimir Muratovski,
Strahil Sokolov

Time Accounting Artificial Neural Networks for
Biochemical Process Models .............................................. 181
Petia Georgieva, Luis Alberto Paz Suárez, Sebastião Fêyo de Azevedo
Contents

Decentralized Adaptive Soft Computing Control of Distributed Parameter Bioprocess Plant .......................... 201
Ieroham S. Baruch, Rosalba Galvan-Guerra

Effective Mutation Operator and Parallel Processing for Nurse Scheduling .............................................. 229
Makoto Ohki, Shin-ya Uneme, Hikaru Kawano

Case Studies for Genetic Algorithms in System Identification Tasks ...................................................... 243
Aki Sorsa, Riikka Peltokangas, Kauko Leiviskä

Range Statistics and Suppressing Snowflakes Detects for Laser Range Finders in Snowfall ................... 261
Sven Rönnbäck, Åke Wernersson

Situational Modelling for Structural Dynamics Control of Industry-Business Processes and Supply Chains .......... 279
Boris Sokolov, Dmitry Ivanov, Alexander Fridman

Computational Study of Non-linear Great Deluge for University Course Timetabling ............................... 309
Joe Henry Obit, Dario Landa-Silva

Entropy Operator in Macrosystem Modeling .......................................................................................... 329
Yu S. Popkov

Generalized Net Model for Parallel Optimization of Feed-Forward Neural Network with Variable Learning Rate Backpropagation Algorithm with Time Limit ............................................................. 361
S. Sotirov, K. Atanassov, M. Krawczak

Towards a Model of the Digital University: A Generalized Net Model for Producing Course Timetables and for Evaluating the Quality of Subjects ................................................................. 373
A. Shannon, D. Orozova, E. Sotirova, M. Hristova, K. Atanassov, M. Krawczak, P. Melo-Pinto, R. Nikolov, S. Sotirov, T. Kim

Intuionistic Fuzzy Data Quality Attribute Model and Aggregation of Data Quality Measurements ............. 383
Diana Boyadzhieva, Boyan Kolev

Redundancy Detection and Removal Tool for Transparent Mamdani Systems ....................................... 397
Andri Riid, Kalle Saastamoinen, Ennu Rüstern
Optimization of Linear Objective Function under Fuzzy Equation Constraint in BL- Algebras – Theory, Algorithm and Software .................................................. 417
Ketty Peeva, Dobromir Petrov

Electric Generator Automation and Protection System
Fuzzy Safety Analysis .............................................. 433
Mariana Dumitrescu

A Multi-purpose Time Series Data Standardization Method .............................................. 445
Veselka Boeva, Elena Tsiporkova

Classification of Coronary Damage in Chronic Chagasic Patients .............................................. 461
Sergio Escalera, Oriol Pujol, Eric Laciar, Jordi Vitrià, Esther Pueyo, Petia Radeva

Action-Planning and Execution from Multimodal Cues: An Integrated Cognitive Model for Artificial Autonomous Systems .............................................. 479
Zenon Mathews, Sergi Bermúdez i Badia, Paul F.M.J. Verschure

Design of a Fuzzy Adaptive Controller for Uncertain Nonlinear Systems with Dead-Zone and Unknown Control Direction .............................................. 499
A. Boukroune, M. M’Saad, M. Tadjine, M. Farza

An Approach for the Development of a Context-Aware and Adaptive eLearning Middleware .............................................. 519
Stanimir Stoyanov, Ivan Ganchev, Ivan Popchev, Máirtín O’Droma

New Strategies Based on Multithreading Methodology in Implementing Ant Colony Optimization Schemes for Improving Resource Management in Large Scale Wireless Communication Systems .............................................. 537
P.M. Papazoglou, D.A. Karras, R.C. Papademetriou

Author Index .............................................. 579
An Approach for the Development of a Context-Aware and Adaptive eLearning Middleware*

Stanimir Stoyanov, Ivan Ganchev, Ivan Popchev, and Máirtín O’Droma

Abstract. This chapter describes a generic, service-oriented and agent-based approach for the development of eLearning intelligent system architectures providing wireless access to electronic services (eServices) and electronic content (eContent) for users equipped with mobile devices, via a set of InfoStations deployed in key points around a University Campus. The approach adopts the ideas suggested by the Model Driven Architecture (MDA) specification of the Object Management Group (OMG). The architectural levels and iterations of the approach are discussed in detail and the resultant context-aware, adaptive middleware architecture is presented. The classification and models of the supporting agents are presented as well.

1 Introduction

One of the main characteristics of the eLearning systems today is the ‘anytime-anywhere-anyhow’ delivery of electronic content (eContent), personalized and customized for each individual user [1], [2]. To satisfy this requirement new types of context-aware and adaptive software architectures are needed, which are enabled to sense aspects of the environment and use this information to adapt their

* The authors wish to acknowledge the support of the Bulgarian Science Fund (Research Project Ref. No. ДО02-149/2008) and the Telecommunications Research Centre, University of Limerick, Ireland.

Stanimir Stoyanov
Department of Computer Systems, Plovdiv University “Paisij Hilendarski”,
Plovdiv, Bulgaria

Ivan Ganchev · Máirtín O’Droma
Telecommunications Research Centre, University of Limerick, Ireland

Ivan Popchev
Bulgarian Academy of Sciences

V. Sgurev et al. (Eds.): Intelligent Systems: From Theory to Practice, SCI 299, pp. 519–535.
springerlink.com © Springer-Verlag Berlin Heidelberg 2010
behavior in response to changing situation. In conformity with [3], a context is any information that can be used to characterize the situation of an entity. An entity may be a person, a place, or an object that is considered relevant to the interaction between a user and an application, including the user and the application themselves.

One of the main goals of the Distributed eLearning Centre (DeLC) project [4], [5] is the development of such an architecture and corresponding software that could be used efficiently for on-line eLearning distance education. The approach adopted for the design and development of the system architecture is of essential importance for the success of this project. Our approach is focused on the development of a service-oriented and agent-based intelligent system architecture providing wireless access to electronic services (eServices) and eContent for users equipped with mobile devices, via a set of InfoStations deployed in key points around a University Campus. The approach is based on the ideas suggested by the Model Driven Architecture (MDA) of the Object Management Group (OMG) [6].

This chapter provides a general description of our approach including its architectural levels and iterations. A context-aware and adaptive middleware architecture developed as a result of this approach is presented. Furthermore the classification and models of the supporting agents are presented as well.

2 InfoStation-Based Network Architecture

The utilized InfoStation-based network architecture provides wireless access to eServices and eContent for users equipped with mobile devices, via a set of InfoStations deployed in key points around a University Campus [7], [8]. The InfoStation paradigm is an extension of the wireless Internet as outlined in [9], where mobile clients interact directly with Web service providers (i.e. InfoStations). The InfoStation-based network architecture consists of the following basic building entities as depicted in Figure 1: user mobile devices (mobile phones, PDAs, laptops/notebooks), InfoStations, and an InfoStation Center (ISC).

The users request services (through their mobile devices) from the nearest InfoStation via available Bluetooth, WiFi/WLAN, or WiMAX/WMAN connections. The InfoStation-based system employs the principles of the distributed control, where the InfoStations act as intelligent wireless access points providing services to users at first instance. Only if an InfoStation cannot fully satisfy the user request, the request is forwarded to the InfoStation Center, which decides on the most appropriate, quickest and cheapest way of delivering the service to the user according to his/her current individual location and mobile device’s capabilities (specified in the user profile). The InfoStation Center maintains an up-to-date repository of all profiles and eContent. The InfoStations themselves maintain cached copies of all recently accessed (and changed) user profiles and service profiles, and act also as local repositories of cached eContent.
3 DeLC Approach

For the development of our eLearning system we use a software development approach based on some fundamental OMG-MDA [6] ideas by taking into account the specifics of the InfoStation infrastructure.

3.1 Model Driven Architecture (MDA)

The development of architectures and information systems satisfying the requirements of the modern eLearning distance education is a complex and sophisticated process. Decompositional approaches could facilitate this process better and lead to faster overall problem solution by allowing a complicated problem to be decomposed into simpler sub-problems. Each sub-problem solution could then be designed and implemented as a separate system component. However, integration of components and common control/management are required in order to realize the total functionality of the entire system. Another factor, which influences the complexity of the modern software design, is the re-usability of the existing components. MDA offers a suitable approach for coping with this situation. The approach is based on the OMG modelling standards that allow systematically to encompass and understand the entire problem before solving it. The MDA approach for the implementation of complex applications is presented in this subsection.

The core of our architecture is based on the following modelling standards proposed by OMG:

- **Unified Modeling Language (UML)** [10]: UML allows models to be created, considered, developed, and processed in a standard way starting from the initial analytical phase and going through all phases up to the final design and development. The UML models allow an application to be developed, assessed, and evaluated before starting to write the actual code. This way all necessary changes in the application could be made much easier and the cost of the overall design process could be reduced significantly;
- **Meta-Object Facility (MOF):** MOF is a standardized way for managing the models using repositories;
- **Common Warehouse Meta-model (CWM):** CWM standardizes the models representation as databases, schemes of transformational models, On-Line Analytical Processing (OLAP) and data mining models, etc.

The core models can be specified in a form of UML profiles. The core models are independent of the middleware platform used. Their number is relatively small because each of them presents features that are common for a particular category of problems. In this sense, the core models are meta-models for different categories. Three types of core models are used:

- **Models of business applications** with component architecture and transactional behaviour;
- **Models of real-time systems** with special requirements for resource control and management;
- **Models of other specialized systems.**

MDA allows applying a common standardized approach for the development of applications independently of the objective middleware platform used. The approach is based on the following steps:

- **Creation of a Platform-Independent Model (PIM)** in UML;
- **Creation of a Platform-Specific Model (PSM)** by mapping of PIM to the actual platform. PSM truly reflects the semantics of the functionality as well as the semantics of the application run-time. This is still a UML model but presented in one of the UML dialects (profiles), which reflects accurately the technical run-time elements of the target platform;
- **Writing the code of the application** that realizes the specific business logic and operational environment. Different types of code and corresponding configuration files are created for different component-oriented environments, e.g. interface files, definition and configuration files, files with primary code, configuration files for integration, etc.

Two main mappings are considered in the MDA approach for the realization of the alliance (coupling) of different levels (models):

- **Mapping of PIM to PSM:** Specialists (knowing in depth and in detail the requirements of the target platform) map the common model of the application (PIM) to a model, which is pursuant to and reflecting the specifics of the platform. Due to a variety of reasons this process could hardly be automated. Despite that, however, there are some automated tools (e.g. CCM, EJB, and MTS) that may facilitate mainly the mappings to standard platforms. Improvements in the automation of this type of mapping are currently hot research topics because they allow reducing significantly the amount of manually performed work. This is especially true for specialized platforms;
• **Mapping of PSM to a working application:** An automatically generated code is complemented with a manually written code, which is specific for the application.

### 3.2 Architectural Levels

In our case, we want to be able to model functionality of eLearning services independently of the utilized InfoStation network (as PIMs). On the other hand, the services should be deployable for provision (PIM mapping) in an InfoStation environment (PSM). In addition we must take into account yet another circumstance, namely the possible changes in the environment during the operation of the system. These changes have to be detected and identified by the system architecture and their effect on the service provision to be taken into consideration. To achieve this, here we propose a more sophisticated structure of PSM, which encompasses the InfoStation environment and the middleware needed for ensuring the required architecture's awareness. The middleware is developed independently as much as possible of the technical details and specifics of the InfoStation network.

Our approach envisages the existence of three architectural levels presented in the next subsections and depicted in Figure 2.

![Diagram](image)

**Fig. 2** DeLC approach: architectural levels and iterations
3.2.1 eLearning Services Level

This level represents and models the functionality of the eLearning services provided by the system as specified in the eLearning Framework (ELF) [12]. ELF is based on a service-oriented factoring of a set of distributed core services required to support eLearning applications, portals and other user agents. Each service defined by the framework is envisaged as being provided as a networked service within an organization.

The service functionality is modelled in UML by taking into account the fact that the service realization is not directly unfolded by the system software but rather is processed by the middleware. The middleware acts as a kind of a virtual machine for the eLearning services. That's why it is very important to present the service as a composition of smaller parameterized activities, which could be navigated in different way. The actual navigation and parameterization depend on the environmental changes identified by the middleware during the provision of the corresponding service.

3.2.2 Middleware Level

This is an agent-based multi-layered level playing a mediator role between the services level and the scenarios level. It offers shared functionality: on one hand, it contains agents needed for the execution of different scenarios; on the other hand, it specifies a set of agents assuring the proper provision of eLearning services. In the light of the MDA philosophy, this level could be considered as PSM, which delivers a virtual (software) environment for service provision.

The main goal of the middleware level is to allow the architecture to execute/satisfy the user requests for eLearning services in a context-aware fashion. The two main tasks related to this are:

1) Detection and identification of all important environmental changes, i.e. the delivery of the relevant context for the provision of the requested services;

2) Adaptation of the architecture (in correspondence to the delivered context) as to support the provision of the requested services in the most efficient and convenient way.

3.2.3 Scenarios Level

This level presents the features and specifics of the InfoStation infrastructure in the form of different scenarios executed for the provision of eLearning services. The main task of the scenarios is to make transparent to the middleware level all the hardware characteristics of network nodes and details of communication in the InfoStation network. Scenarios reflect the main situations that are possible to happen in the InfoStation environment and related to the main task of the middleware, i.e. ensuring context-aware execution of user requests for eLearning services. Due to device mobility (i.e. moving between geographically intermittent InfoStation cells) and user mobility (i.e. shifting to another mobile device) the following four basic scenarios are possible [11]:


1) **'No change' scenario**: If the local InfoStation can fulfil the user service request, the result is returned to the user. However, if the InfoStation is unable to meet the demands of the user, the request is forwarded onto the InfoStation Center, which retrieves the required eContent from a repository and sends it back to the InfoStation. The InfoStation may reformat/adapt the eContent in accordance with the user profile and then sends the adapted eContent to the user mobile device. The InfoStation also stores a copy the new eContent in its cache, in case if another user requests the same eContent.

2) **'Change of device' scenario**: Due to the user mobility, it is entirely possible that during a service provision, the user may shift to another mobile device. For instance, by switching to a device with greater capabilities (for example from a PDA to a laptop), the user may experience a much richer service environment and utilize a wider range of resources. In this scenario, the mobile device sends a notification of device change to the InfoStation, detailing the make and model parameters of the new device. Then the InfoStation reformat the service eContent into a new format, which best suits the capabilities of the new user device.

3) **'Change of InfoStation' scenario**: Within the InfoStation paradigm, the connection between the InfoStations and user mobile devices is by definition geographically intermittent. With a number of InfoStations positioned around the University Campus, the users may pass through a number of InfoStation cells during the service session. This transition between InfoStation cells must be completely transparent to the user, ensuring the user has apparent un-interrupted access to the service. As the user moves away from the footnote (service area) of an InfoStation, the user mobile device requests user de-registration from the current InfoStation. The device also requests one last user service profile update before leaving the coverage area of the current InfoStation. The InfoStation de-registers the user, updates the cached profile, and forwards the profile update to the InfoStation Center to make necessary changes in the Master Profile Repository. Meanwhile the execution of the user’s request continues (for example reading through the downloaded eContent or completing the tests at the end of lecture’s sections). When the user arrives within the coverage area of another InfoStation, the service execution continues from the last (synch) point reached by the user.

4) **'Change of device & InfoStation' scenario**: We have outlined the separate instances where the user may switch his/her access device or pass between InfoStation cells during a service session. However, a situation may arise where the user may change the device simultaneously with the change of an InfoStation. In this scenario, both procedures for device change and InfoStation change may be considered as autonomic procedures, independent of each other. Hence each of these may be executed and completed at any point inside the other procedure without a hindrance to it.
3.3 Iterations

A flexible approach is needed for the development of a context-aware and adaptive architecture in order to examine different development aspects and be able to extend the architecture step by step. The main idea behind our approach is to consider the system development as a process of iterations. The term iteration – borrowed from the Unified Software Development Process [13] – means a workflow or cooperation between the developers at different levels so as to be able to use and share particular products and artefacts.

There are two distinguished types of iterations in our approach (Figure 2):

- **SM iterations** - between the scenarios level and the middleware level. During each SM iteration, new scenarios that present/reflect particular aspects of the possible states and changes in the environment are developed (or the existing scenarios are modified and/or re-developed in more details). This way using the (formalized) presentation of scenarios, all corresponding middleware components needed for the support of these scenarios are developed step by step.

- **eLM iterations** - mappings of the eLearning services onto the middleware level, where the navigation model and parameterization of services are specified.

For the middleware development we plan the realization of six main SM iterations as described in the following subsections.

3.3.1 Basic Architecture

The first iteration aims at the development of the basic scenarios (presented in the previous subsection), which reflect the main changes that may happen in the InfoStation-based environment due to device mobility and user mobility. Based on these scenarios, a basic eLearning architecture has been developed. This architecture is presented in more detail in the next section.

3.3.2 Time-Based Management

Some important changes in the context during the user service request’s execution – e.g. the device mobility – can be detected and identified by the system only if the temporal aspects of this process are taken into consideration. Thus the goal of this iteration is to develop concepts and formal models allowing a temporal adaptation of the processes supported by the middleware.

3.3.3 Adaptation

This iteration is concerned with problems related to strengthening the architecture, e.g. to support adaptability. In our opinion, personalized eLearning could be fully realized only by means of adaptive architectures, whereby the eLearning content is clearly distinguished from the three models influencing the learning process – the user model, the domain model, and the pedagogical model. The user model presents all information related to the learner’s individuality, which is essential for
the realization of a personalized learning process. The domain model presents the structure of the topic/subject for study/learning. In addition, in our architecture we want to support a goal-driven learning process, whereby in case of a learner’s request sent to the system, a concrete pedagogical goal is generated based on the pedagogical model. The entire management of the user session afterwards obeys this pedagogical goal. These three models are supported explicitly in our architecture. They represent a strong foundation for seeking opportunities for adaptation to environmental/context changes so as to ensure more efficient personalized learning (in this sense we aim at realization of a user/domain/pedagogical model-driven optimization).

3.3.4 Resource Deficit
In some cases the user requests for particular services cannot be satisfied fully by the local InfoStation due to resource deficit (e.g. when information needed to satisfy the service request is unavailable in the database of the local InfoStation). In these cases the service provision must be globalized in a manner involving other InfoStations (through the InfoStation Center). The software needed to support this type of InfoStations interaction is developed as part of this iteration. The resource deficit in the serving InfoStations is caused not by dynamic factors but rather by the static deployment of resources on network nodes.

3.3.5 Collaboration
In many cases the execution of particular service requests requires interaction between the middleware agents. Usually information that is needed for making the decision is gathered locally, whereas the decision must be made by means of communication, cooperation, and collaboration between agents (centralized management of electronic resources and services is not envisaged in DeLC). During this iteration, the development of a common concept, models and supporting means for both local (within the service area of an InfoStation) and global (within the entire InfoStation network) agents’ collaboration is envisaged.

3.3.6 Optimization
This iteration investigates the possibilities for optimal functioning of the middleware and proposes relevant corrections and extensions to the architecture. Different possibilities in the proposed InfoStation-based infrastructure exist for seeking the optimal solutions, e.g., the development of intelligent agents with new abilities (cloning, copying, mobility), which could be used to balance the workload out on the network nodes.

4 Basic System Architecture

This section presents the basic architecture, which was developed during the first SM iteration described in the previous section.
4.1 Tiers and Layers

In keeping with the principles of the InfoStation network, which can support a context-aware service provision, we develop a software for the three tiers of the architecture, namely for the mobile devices, InfoStations, and InfoStation Center [14]. In the standard InfoStation architecture, mobile devices use InfoStations only as mediators for accessing the services offered by the InfoStation Center. In our concept we foresee the spreading role of the InfoStations, which (besides the mediation role) act as hosts for the local eLearning services (LeS) and for preparation, adaptation, and conclusive operations of global eLearning services (GeS). This way the service provision is distributed across the whole architecture in an efficient way. The layered system architecture is depicted in Figure 3.

![Layered System Architecture Diagram]

**Fig. 3** The layered system architecture

Different phases of a particular service provision may be carried out on different tiers of the architecture according to the scenario, which is currently executed. Mobile devices are provided with wireless access to services, offered by the InfoStations and/or InfoStation Center. Conceptually, the architecture required for maintaining the InfoStation configurations, is decomposed into the following logical layers: communications layers (Ethernet, mobile communications – MoCom, TCP/IP), middleware layer, service interface layer, and service layer (Figure 3). The middleware layer is responsible for detecting and identifying all the changes in the environment that may affect the provision of services requested by users and for the relevant adaptation of the architecture in response to these changes (i.e. this layer supports the context-awareness aspect of the architecture). The service layer selects and activates the requested service. Details of the middleware layer are presented in the next subsections.
4.2 Middleware Agents

In order to facilitate the context-aware service provision, the middleware consists of different intelligent agents (deployed at different tiers of the InfoStation network), which interact to each other so as to satisfy in the 'best' possible way any user request they might encounter. Here we present the different types of middleware agents operating on different nodes of the InforStation network (i.e. on user mobile devices, InfoStations, InfoStation Center). The classification of the agent types is presented as an Agent-based Unified Modeling Language (AUML) diagram [15]. (Figure 4). AUML is an extension of UML and is used for modelling agent-based architectures.

![Middleware Agent Classification Diagram](image)

Fig. 4 The middleware agent classification.

The functionality of each class of middleware agents is described in the next subsections.

4.2.1 Personal Assistant Class
This class encompasses the personal assistants installed on the user mobile devices (smart phones, PDAs, laptops, etc). The task of these agents is to help users request and use different services when working with the system.

4.2.2 Communicator Class
The task of this class of agents is to provide communication between different tiers of the InfoStation architecture. The main types of wireless communication used within the InfoStation environment are Bluetooth, WiFi, and WiMAX. Separate agents are developed for each of these. In addition, in accordance with the Always Best Connected and best Served (ABC&S) communication paradigm [20, 21], ABC&S agents help to choose (transparently to users) always the best connection available for each particular service requested by the user in each particular moment depending on the current context (e.g. the noise in communication channel, error rate, network congestion level, etc.).

The model of the Bluetooth communication agent is presented here as an example. This agent helps discovering the services, searching for other Bluetooth
devices, establishing a connection with them, and detecting any lost of connection (e.g. out-of-radio-range problem). The main class here is the BluetoothBehaviour class (Figure 5).

Additional classes are:

- **MessageSpeaker, MessageSpeakerSomeone** – used to send messages to one or many agents simultaneously;
- **ParticipantsManager** – used to receive up-to-date information about other agents currently available in the InfoStation environment;
- **MessageListener** – with an ability to capture messages from a particular type of agents bound to the same InfoStation;
- **CyclicBehaviour, OneShotBehaviour** – used for the realization of the cyclic behaviour and the one-shot behaviour, respectively;
- **ConnectionListener** – an interface helping to track the status of the connection with the JADE platform [18] (more precisely the connection between Front Ends and Back Ends);
- **MyBIFEDispatcher** – supports the IP communication between different InfoStations;
- BluetoothClientAgent – offers different methods needed for the maintenance of personal assistants, e.g. initial setup, end of work (takedown), different behaviour activations (handleSpoken, handleSpokenSomeone), processing of list of agents registered on a particular InfoStation, etc.

4.2.3 Manager Class
The agents in this class ensure proper detection and identification of all important events in the environment (e.g. events related to device mobility and user mobility) and deliver the actual context for the execution of the requested service. In doing this, the agents take into account the time characteristics of the events.

4.2.4 Adapter Class
These agents ensure the necessary adaptability of the architecture in response to the provided context from the manager agents. The adaptation model distinguishes two main groups of artifacts:

- Adaptation objects: These are defined data structures that must be changed in a certain way (depending on adaptation subjects) before being offered to the users. The three main types of adaptation objects are: content, domain, and service;
- Adaptation subjects: These are the system users and their mobile devices. These are sources for different limitations/restrictions towards the adaptation objects. The restrictive conditions of the subjects are generalized and presented as profiles. Two main profiles are supported – user profiles and device profiles. Using the information stored in these profiles, the eContent can be adapted and customized according to the user preferences and capabilities of the user device currently in use. For instance, the user mobile device (a cellular phone) may be limited in its capabilities to play video content in which case video components are sent in a format that best suits the device, or they may be simply omitted. The user may choose to access the full capabilities of the eContent later, when using a device with greater capabilities (e.g. a laptop).

The Adapter class consists of two subclasses - Subject class and Profiler class. The Subject class provides three specialized agent types respectively for adaptation of: content (Content class), courses/modules (Domain class), and eLearning services (Service class). The Profiler class utilizes the “Composite Capabilities/Preference Profile” (CC/PP) standard [16]. The Master Profile repository in the InfoStation Center contains descriptions of all registered user mobile devices, i.e. their capabilities and technical characteristics. During the initialization, the user’s personal assistant sends as parameters the make and model of the user device. An agent working on the InfoStation (or the InfoStation Center) reads the corresponding device’s description from the repository and according to this, selects the ‘best’ format of the eContent, which is then
forwarded to the user. For the support and processing of profiles we use two separate agent classes – Device class and User class.

4.2.5 Collaborator Class
Collaboration (like adaptation) must be designed and built into the system from the start [17]; it cannot be patched on. This special agent class is required for the support and control of the run-time collaboration model. Besides the specification of agents needed for the support of the possible scenarios’ execution, a specification of the possible relationship and interactions between agents is also needed. The agent collaboration has the potential to enhance the effectiveness of teamwork within the DeLC infrastructure. The roles played by the participants in a collaborative eLearning activity are important factors in achieving successfully the learning outcomes.

4.2.6 Service Communicator Class
The main designation of these agents is to provide an interface to the services that represent the main system functionality. This class of agents realizes the service interface layer in Figure 3.

4.3 Middleware Architecture
As mentioned before, during the first SM iteration a basic version of the InfoStation’s middleware architecture was developed and implemented as a set of cooperating agents (Figure 6). The agents perform different actions such as: searching for and finding mobile devices within the range of an InfoStation, creating a list of services required by mobile devices, initiation of a wireless connection with mobile devices, data transfer to- and from mobile devices, etc.

Fig. 6 The InfoStation’s middleware architecture
A short description of different agent types is provided in the subsections below.

4.3.1 Scanner Agent
This agent searches and finds mobile devices (within the range of an InfoStation) with enabled/activated wireless interface corresponding to the type of the InfoStation (Bluetooth/WiFi/WiMAX or mixed). In addition, this agent retrieves a list of services required by users (services are registered on mobile devices upon installation of the client part of the application and started automatically by the InfoStation agents).

4.3.2 Connection Adviser Agent
The main task of this agent is to filter the list (received from the Scanner agent) of mobile devices and services. The filtration is carried out with respect to a given (usually heuristic) criterion. Information needed for the filtration is stored in a special database (DB). The Connection Adviser agent sends the filtered list to the Connection Initiator agent.

4.3.3 Connection Initiator Agent
This agent initiates a communication required for obtaining the service(s) requested by the user. This agent generates the so-called Connection Object, through which a communication with the mobile device is established via Bluetooth or WiFi or WiMAX connection. In addition, for each active mobile device it generates a corresponding Connection agent, to which it handovers the control of the established wireless connection with this device.

4.3.4 Connection agent
The internal architecture of this agent contains three threads: an Agent Thread used for communication with the Query Manager agent, and a Send Thread and a Receive Thread, which support a bi-directional wireless communication with the mobile device.

4.3.5 Query Manager Agent
This agent is one of the most complicated components of the InfoStation’s architecture. On one hand, the Query Manager prepares and determines where information received from the mobile device is to be directed, e.g. to simple services, or to sophisticated services via Interface agents. For this purpose, this agent transforms the messages coming from the Connection agent into messages of the corresponding protocols, e.g. UDDI or SOAP for simple services. For direct activation of simple services (e.g. Web services) there is no need for Interface agents. The latter are designed to maintain communication with more complicated services by using more sophisticated, semantic-oriented protocols (e.g. OWL-S [19]). In this case, the Query Manager acts as a mediator. In the opposite direction, this agent transforms the service execution’s results into messages understandable by the agents. This operation is needed because
results must be returned to the relevant *Connection* agent, which has requested the provision of the service on behalf of the user.

### 4.3.6 Scenario Manager Agent
This agent performs the time-based scenario management based on a suitable way for formalizing the scenario presentation. For instance, the *Change of InfoStation* scenario could be specified mainly through the user (device) movement from one InfoStation to another. To detect this event, however, two local events must be detected during the run-time: (i) the user/device leaving the service area of the current InfoStation; (ii) the same user/device entering the service area of another InfoStation. The scenario identification cannot be centralized due to the necessity to detect these two local events. This is achieved through a message exchange between the two *Scenario Manager* agents running on the two InfoStations. The messages include the start time of events, mobile device’s identification, and other parameters.

### 5 Conclusion
This chapter has presented an approach for the development of service-oriented and agent-based eLearning intelligent system architectures supporting wireless access to electronic services (eServices) and electronic content (eContent) for users equipped with mobile devices, via a set of InfoStations deployed in key points around a University Campus. The approach adopts the ideas suggested by the Model Driven Architecture (MDA) specification of the Object Management Group (OMG). The architectural levels and iterations of the approach have been discussed in detail. The first version of the resultant context-aware and adaptive middleware architecture developed accordingly to this approach by means of the agent-oriented platform JADE has been presented.

### References


[18] JADE - Java Agent DEvelopment framework (to date), http://jade.cselt.it

