9

Semantically Enriched Integration Framework for Ubiquitous Computing Environment

Habib Abdulrab, Eduard Babkin and Oleg Kozyrev
[1]*LITIS laboratory, INSA de Rouen*
[2]*State University – Higher School of Economics*
[1]*France*
[2]*Russia*

1. Introduction

Miniaturization, reduced costs of electronic components, and advanced information technologies now open practical possibilities to design, develop and deploy thousands of the coin-sized sensors and mechanical devices at multiple locations. This kind of software-hardware systems, pervasively available to the user in everyday activities, is named Ubiquitous Computing Environment (UCE) (Abowd & Mynatt, 2000; Niemelä & Latvakoski 2004), or even - Ubiquitous Smart Space (Jeng, 2004 ; Kawahara et al., 2004). Establishing *ad hoc* communication via wireless media numerous elements of the UCE provide the user with real-time global sensing, context-aware informational retrieval, and enhanced visualization capabilities. In effect, they give extremely new abilities to look at and interact with our habitat. Many researches made a contribution to developing of Sensors and Actuators Networks (SANET), which became a foundation of UCE. There are tiny hardware devices available in practice for building SANET, embedded operating systems, wireless network protocols, and algorithms of effective energy management (Misc.Tinyos, 2010; Feng et al., 2002; Tilak et al., 2002; Crossbow, 2010). Now researchers' community demonstrates growing interest to resolving the next important problem that will be faced by the developers and the users of UCE since a short time. That is the problem of semantic interoperability in the joint context of SANET, existing IT-infrastructure and people society.
Resent results (Branch et al., 2005 ; Curino et al., 2005 ; Tsetsos et al., 2005; Ahamed et al., 2004; Tokunaga et al., 2004 ; Chan et al., 2005) show applicability of the middleware paradigm for the solution of that problem, and provide for approaches facilitating integration of SANET on the application level of enterprise systems.
However in the case of actual wide-area UCE, multiple SANETs spread across administrative borders, enterprises, and even social cultures. Deep involving of tiny computing devices in our everyday activities requires closer coincidence of computer interfaces with people's way of perception and mental world models. As activities and social experience are different, the mental world models also differ. So, interfacing with the same sensors can be absolutely dissimilar in respect with the style, modality and informational contents. The same raw data collected by sensors can be interpreted differently and can be applied in absolutely divergent contexts. This simple fact breaks a "closed world" assumption, and requires shedding light of researcher's attention on such

issues as explicit meta-data representation, and formal modelling of system properties and interfaces to achieve semantic interoperability.
In our research we explore ways to extend existing partial middleware solutions in UCE with a consistent model-driven methodology for semi-automated design and development of semantic integration components called "Ontology Mediators". The main purpose of Ontology Mediator is communication with SANETs, data integration and seamless fusion of diverging real-world concepts and relationships in accordance with information needs of certain single user or a small user's group. Depending on specific conditions and requirements, implementation of Ontology Mediator varies from specialized middleware components to reconfigurable hardware devices. Tailored for local *ad hoc* requirements Ontology Mediator provides strong support for the claim (Herring, 2000) "…that computer products now eventually progress from large, general-purpose, impersonal static forms to portable, personal, flexible, market-targeted forms. Personalization, flexibility, and quick time to market dictates a 'quick turn' design approach."
During domain modelling, design, and development of Ontology Mediators different end-user tools, component libraries and algorithms should be used. No doubt, the best results can be achieved when all these elements are combined into the vertical framework supporting all stages of the methodology. We have designed architecture of such semantic interoperability framework suitable for loosely coupled distributed systems like SANETs, and developed a number of software and hardware prototypes to evaluate benefits and afford proof of Ontology Mediator and the design methodology. This framework supports semi-automated design and development of Ontology Mediators, as well as it allows for designing and establishing coherent information flow between different components on the basis of coordinated ontology transformation activities. In the course of the prototype implementation we applied RDF-model transformations in order to provide semantic interoperability and semantic validation, and explored different kinds of software technologies (the JavaSpace, JMS Messaging, and CORBA). Altogether, these contributions are used for rapid development of highly customized Ontology Mediators in UCE.
In this work we describe most important characteristics of the proposed framework as follows. In Section 3 our motivation is explained using a specific use case of semantic integration. Section 3 gives a short description of foundations and relevant topics to our research. Section 4 contains explanation of major steps in our methodology of Ontology Mediator's design. The general architecture of the supporting framework is presented in Section 5. Sections 6 and 7 give a detailed view on the most important components of the framework: the Transformation Engine (T-Engine) and the extensible hardware platform. We summarize obtained results and compare them with other known approaches in the conclusion (Section 8).

2. Motivation

In order to support necessity of a specific methodology for design and development of Ontology Mediators we propose to concern a case study of their application in the context of wide-area UCE. In this case study modern seaports were chosen for consideration due to significant role of sea transportation in economics and its great impact on environmental safety and security.
Since last thirty years seaport infrastructure became an extremely complex system where multiple physical objects with interfering properties, abstract logical concepts and

normative procedures are tightly coupled to support 24-hour cargo operations, transportation logistics, custom and security checking. Although a usual seaport provides for different services like containers import-export, oil terminals and passengers transportation, the former kind of services plays a major role. Different authors estimate amount of container operations from 80 to 90 percent of total world cargo throughput. And most of these operations are concentrated at a relatively small number of huge ports. For example, in 2003 the total European container throughput was 50 million TEUs (Twenty-foot Equivalent Units, standard container volume measure); more than half of containers were processed on the Hamburg-Le Havre range ports (25.4 million TEU). It is expected that over the next 20 years, demands on seaport capacity will double. However, most major seaports cannot grow larger, so increase in capacity and port productivity can be achieved mostly in result of elaborate business process reengineering and broad application of advanced IT-solutions which obviously include UCE.

To show applicability of UCE and needs for continuously evolving mediation of ontologies we concern a typical port with berths at the dock facility, a container terminal with gantries, a container yard with cranes and forklifts, an oil terminal, an extensive land transportation network including railroads and truck routes and gates. In such complex heterogeneous environment IT-infrastructure of the seaport should support continuous operations of different own and third party employees, effective supply chain management, logistic and highest level of security within maritime and ground port areas as well as surrounding territories. Apart from internal client-server or service-oriented information systems IT-infrastructure also includes external information sources and three different kinds of SANETs spreading across the seaport territory and nearest regions. The sensors of the first SANET (SANET#1) monitor such environmental conditions as temperature, humidity, biological and chemical contamination levels. The sensors of the second network (SANET#2) precisely track container and other objects movement with the help of RFID and GPS technologies within the seaport. The third network (SANET#3) supports surveillance, personal identification and access control. At last the wide-area sensors network (SANET#4) gathers information about traffic conditions for the main regional routes.

Among multiple participants of seaport business-processes we select only three groups of individuals with specific informational interests for further analysis and assign to them certain roles: the truck driver (TD-User), the security officer (SO-User), and the manager of the long-distance goods delivery service (DM-User). Table 1 contains the key issues each user usually faces in the context of business activities.

Except differences in the world models the users also employ different software application models. DM-USER has a stable location and primarily uses a desktop application with service-oriented architecture. SO-USER and TD-USER are equipped with mobile special terminals; we can say that they are “immersed” into the UCE and should directly interact with the sensor networks.

Although the world concepts are seemed to be absolutely different for three concerned users, construction of these concepts requires access to the same sensors measurements at the lowest level of abstraction.

For example, the world model of TD-USER consists of such high-level concepts as distances, container dimensions, speed limits, map directions, goods damage risks. For such the model raw RFID and GPS data acquired by SANET#2 should be transformed, analytically processed and integrated with traffic and weather condition information of SANET#1 and SANET#4. While TD-USER remains inside the sea port the sensors of SANET#3 provide for

TD-USER	The shortest and safe route to the uploading position and estimated queuing time. Average speed of transportation with respect to road conditions, traffic level and specifics of the goods to be transported. Import-export declaration for transported goods.
SO-USER	Access privileges for a particular area in the port and privileges assignment policies. Destination and intermediate checkpoints of vulnerable goods. Location of the next container for security inspection and the list of necessary screening and inspections procedures. Possible threats and security risks with respect to global security alerts, the current situation in the port and weather conditions. Where in some proximity to a given location specific goods can be found. Impact of certain kinds of cargo on nearly located materials and goods. Consequences of port accidents and their influence on surrounding areas. Risk mitigation routines adapting to the actual situation and environmental conditions.
DM-USER	Estimated quality of goods in accordance with past and present storage conditions, duration of travel and other circumstances. The list of the goods that can be shipped or stored together to reduce costs of operations. Probable delays in cargo operations due to night-time, weather or other restrictions. Average time of delivery to the city local storage, demands of remote customers, estimations of supply levels, schedules of connection between local carriers and airlines.

Table 1. Subjects of interest for different groups of the users

access control information needed for quickest route determination. For SO-USER following concepts are necessary: trucks and ships entering the port, results of radiation, chemical, biological monitoring, oil leakage facts, loading level of oil terminals, number of tankers at the berths, weather conditions, and permissions violation accidents. For building this model measurements of SANET#1, SANET#2 and SANET#3 as well as external security information sources are necessary. Finally DM-USER needs aggregated information in terms of storage conditions from SANET#1 to predict critical delivery dates. At the same time that user performs analysis of delivery delays in terms of maximum safe truck speed and cargo operations limitations (e.g. oil change delays, capacity of oil terminal). These characteristics can be calculated on the basis of operational data from information systems, local weather conditions and traffic level measured by SANET#1 and SANET#4.

The described conceptual models and data representation formats are very specific and sensitive to peculiarities of user's activities. Proper implementation of these models requires seamless integration of multiple heterogeneous software and hardware components operating within different restrictions (e.g. user interfaces and protocols supported, failover capabilities, battery life, protection class, etc). At the same time continuous and uncorrelated changes in legal regulations, regional environment, seaport business processes and underlying infrastructure lead to permanent evolution of integration algorithms, data structures and communication technologies.

The later condition makes impractical development of a centralized middleware system responsible for data acquisition and semantic transformation: each time when the user's requirements are changed the structure and algorithms of the system are changed dramatically. Maintenance and integration costs of the centralized approach overcome practical abilities. Analyzing another extreme of modern distributed information systems,

the service-oriented architecture, we can point to relatively poor reusability capabilities: for any new combination of hardware and software platform a considerable amount of efforts is needed to design and implement an isolated service, even if mapping principles between different information models are already known on a conceptual level.
In this situation a whole family of semi-automatically generated Ontology Mediators becomes a more convenient solution. The family of Ontology Mediators includes middleware components as well as end-user oriented devices located near sources of raw data. Despite differences in hardware and software design all members of the family of Ontology Mediators have certain common features: conceptual modelling with the same modelling tools, reusable hardware and software components, and similar formal foundations of semantic transformations.
Altogether they play a role of semantic filters that provide only valuable information for the user or for other information systems in terms of the recipient's world model. In our seaport example we can recognize at least four Ontology Mediators supporting users' activities:

- the software Ontology Mediator that feeds the seaport information system with information needed for DM-USER;
- the hardware Ontology Mediator combined with the mobile terminal of TD-USER;
- the hardware Ontology Mediator combined with the mobile terminal of SO-USER.

Design, development and continuous maintenance of Ontology Mediators involve many participants with different skills, roles and administrative responsibilities. We believe that in order to support the described scenario and facilitate rapid replacement, re-design and deployment of different kinds of Ontology Mediators an advanced methodology of automated or semi-automated design should be applied.

3. Formal foundations

From a conceptual point of view our research is related with knowledge engineering and knowledge integration techniques specialized for pervasive computations (Chen et al., 2004; Zhou et al., 2005; Hönle et al, 2005). In this area the concept of ontology plays now the leading role for knowledge representation. It is became a good breeding to refer to the Gruber's pioneer definition of ontology as "a specification of a conceptualization" (Gruber, 1995). According to (Sowa, 2000) ontology serves for strong support in detailed study of all potentially possible entities and their interrelations in some domain of discourse shared by multiple communities; ontology also enables conceptualization and forming categories of the entities committed by those communities. This direct connection of the ontology technique to integration is pointed out by Y. Kalfoglou: "An ontology is an explicit representation of a shared understanding of the important concepts in some domain of interest. (Kalgoglou, 2001)" In (Dragan et al., 2006) one can see a good collection of more recent cross-references, all of them underline ontology support for achieving interoperability: "Ontology … can be seen as the study of the organization and the nature of the world independently of the form of our knowledge about it."
To build the formal foundations for our methodology of Ontology Mediator's design we apply a well-defined and elegant mathematical theory of ontology developed at the University of Karlsruhe (Ehrig, 2007). That theory defines a core ontology (the intentional aspect of the domain of discourse) as a mathematical structure $S = \langle C, \leq_C, R, \sigma, \leq_R \rangle$, where
C – is a set of concept identifiers (concepts for short).
R – is a set of relations identifiers (relations for short).

$\leq_C$ – is a partial order on *C*, called concept hierarchy or taxonomy.
σ – a function $R \rightarrow C \times C$ called signature, such that σ I= ⟨domI, ranI⟩, where $r \in R$, domain domI, range ranI.
$\leq_R$ – is a partial order on *R*, called relation hierarchy, such that $r_1 \leq_R r_2$ if and only if $dom(r_1) \leq_C dom(r_2)$ and $ran(r_1) \leq_C ran(r_2)$.
Domain-specific dependencies of concepts and relations in S are formulated by a certain logical language (e.g. first-order predicate calculus) which fits a rather generic definition:
Let L be a logical language. An L-axiom system for a core ontology is a pair A = <*AI*, α>,where
AI – is a set of axioms identifiers.
α – is a mapping $AI \rightarrow$ L
The elements of A are called axioms.
Extensional definition of the domain of discourse (assertions or facts about instances and relations) is given by description of the knowledge base KB. KB is the following structure:
KB = ⟨*C*, *R*, *I*, ι_C, ι_R ⟩, where
C – is a set of concepts.
R – is a set of relations.
I – is a set of instance identifiers (instances for short).
ι_C – is a function $C \rightarrow P(I)$ called concept instantiation.
ι_R – is a function $C \rightarrow P(I^2)$ called relation instantiation; it has such properties:
$\forall r \in R$, $\iota_R I \subseteq \iota_C$ (domI) $\times \iota_C$ (ranI).
The theory provides also names for concepts and relations calling them signs, and defines a lexicon for ontology:
Lex = ⟨G_C, G_R, G_I, Ref_C, Ref_R, Ref_I ⟩, where
G_C – is a set of concepts signs.
G_R – is a set of relations signs.
G_I – is a set of instances signs.
Ref_C – is a relation $Ref_C \subseteq G_C \times C$ called lexical reference for concepts.
Ref_R – is a relation $Ref_R \subseteq G_R \times R$ called lexical reference for relations.
Ref_I – is a relation $Ref_I \subseteq G_I \times I$ called lexical reference for instances.
In summary, a complete ontology O is defined through the following structure:
O = ⟨*S*, *A*, *KB*, *Lex* ⟩, where
S – is a core ontology.
A – is the L-axiom system.
KB – is a knowledge base.
Lex – is a lexicon.
Such strict mathematical theory of ontology along with other similar approaches formed a solid foundation for machine-readable representation of ontologies in modern information systems and facilitated their practical applications. For example, the research line known as Semantic Web pays great attention to various practical aspects of ontology manipulation for information heterogeneity resolution in the Internet. Recent examples of XML language application in coordination frameworks (Niemelä & Latvakoski, 2004 ; Tokunaga et al., 2004) illustrate that Semantic Web solutions can be successfully adopted and applied in many different domains. It seems for us that potential usefulness of the Resource

Description Framework (RDF) and the RDF Scheme (RDFS) standards (Jeng, 2004), proposed initially for semantic enrichment of WEB contents, is much greater. In order to achieve semantic interoperability for UCE we suggest applying RDF standards as the basic technique for the interoperable description of the knowledge base KB. Fusion of the formal theory of ontology together with RDF allows representing the knowledge base in the single frame of semi-structured data models. In (Abiteboul, 1995) semi-structured data are defined as data that is neither raw, nor strictly typed as in conventional database systems. A convenient approach to represent semi-structured data uses such edge-labeled graph structures which contain both type definition and actual data elements (fig.1).

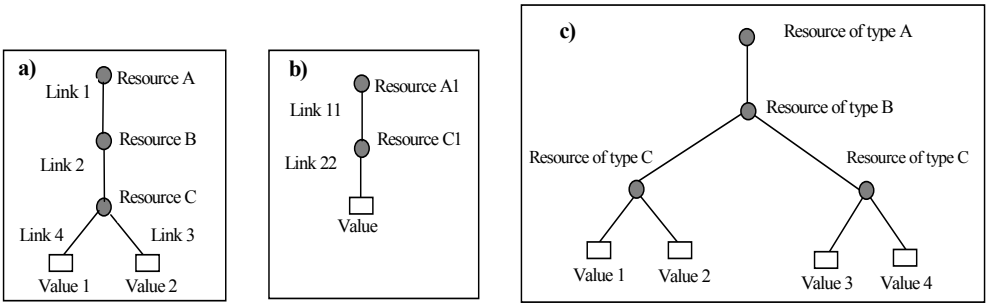

Fig. 1. Three different kinds of semi-structured data

Describing this case, P. Buneman says about "blurring the distinction between schema and instance" and proposes a suitable formalism for modelling and querying such structures (Buneman et al., 1996; Buneman, 1997). In accordance with the Buneman's approach a semi-structured database is represented in a form of a rooted edge-labelled graph, and a schema is a rooted graph whose edges are labelled with formulas. A database SDB conforms to a schema SS if there is a correspondence between the edges in SDB and SS, such that whenever there is an edge labelled a in SDB, there is a corresponding edge labelled with predicate p in SS such that $p(a)$ holds.

We can naturally set a correspondence between elements of the RDF semi-structured database SDB and the knowledge base KB: the set of concepts C and the set of instances I become the nodes of the graph, and the set of relations R maps to the graph's edges. In this context the general problem of achieving semantic interoperability can be looked at as the task of graph-based ontology transformation. The source graph of the semi-structured database comprises messages from the sensors networks, and corresponds to the ontology of IT-infrastructure. Another ontology expresses the user's domain of discourse in terms of the knowledge base, and defines permitted structure of the destination graph. Applying the formal approach and visual cues of graph transformations (Rozenberg, 1997; Hoffmann & Minas, 2000) we may define a transformation workflow process with the aim to produce the destination graph of semi-structured database expressed in terms of user's ontology. The definition of the transformation workflow process consists of two different kinds of operations: the data transformation and the structure transformation. The data transformation uses leaf's values of the source graph, namely the literals of the RDF model, to produce values of the destination graph, namely literals of another RDF model. The data transformation is described as a sequence of interrelated data processing operations that can include elementary RDF literals' transformation functions as well as access operations to

external databases and Commercial Out-of-Shelf Software. The structure transformation uses location information of different subparts of the source graph to build the corresponding subparts of the destination graph.

4. Description of methodology

Concerning implementation and design issues we share the opinion (Oliver, 2005; Volgyesi & Ledeczi, 2002), which says that software engineering knowledge representation, its transformation and automation of systems design can be achieved by application of formal model-driven approaches examining hierarchies of UML-based meta-models, models and ontologies. In our approach to consistent design and development of the Ontology Mediators' family, we apply a few foundation principles for unification of work activities and formal methods. First of all, the consistent three-level UML modelling paradigm is used to create all information models and ontologies. At the top level 'M3' UML Meta-Object Facility provides for a single consistent meta-meta-model; at the middle level 'M2' UML profiles play a role of meta-models and specify domain languages for various aspects descriptions; at the level 'M1' a collection of UML models represents formal description of IT-infrastructure, business view and architecture concepts of Ontology Mediator. The second distinctive feature of our methodology is coupled consideration of classes together with instances during modelling of IT-infrastructure and Enterprise business aspects. It allows for application of formal methods of heterogeneous models and ontologies mapping based on Information Flow theory (Barwise & Seligman, 1997). At the same time simultaneous working with classes and instances facilitates natural application of semi-structured data models. Representation of domain concepts and actual data in the context of the same semi-structured data model gives as an opportunity to employ a powerful technique of graph-oriented transformations in order to define rules of ontology transformation and methods of mapping between different models.

Fig.2. illustrates general principles of the proposed design methodology, in which three main tracks of modelling and design activities can be recognized. All three tracks begin from conceptualization of correspondent domains of discourse. Conceptualization activities of the first track produce UML models with architecture concepts of Ontology Mediator. The architecture models describe reusable software and hardware components and define extensibility interfaces for future use. The second track includes analysis of Enterprise business aspects and producing whole enterprise ontology. Based on Ontology UML profiles, that ontology describes structure and behaviour of the appropriate knowledge domain, expressed in form of appropriate UML concepts (classes and logic constraints). The third track starts from conceptualization of underlying IT-infrastructure in terms of specialized UML models.

Following our methodology the designer should perform Core Activities (conceptualization of IT-infrastructure, Enterprise business aspects and architecture of Ontology Mediator) only for the first time of methodology's application in new domain. But specification of UML-based user's domain ontology is the repetitive activity and it precedes development of any new Ontology Mediator device. In terms of the produced ontology shared instances are classified and matching between enterprise ontology and user's ontology is performed. In the result the developers can select necessary elements of IT-infrastructure to communicate with, and they can design a correspondent source semi-structured data model in terms of IT-infrastructure models. By a similar way, the destination semi-structured data model is

developed in terms of the user's ontology. Once the source and destination models are produced all three tracks are joined, and mutual design of a semantic transformation model is performed. In terms of specialized UML profiles this model describes a workflow transforming the source model to the destination model. In fig.3 a fragment of the correspondent transformation profile is represented.

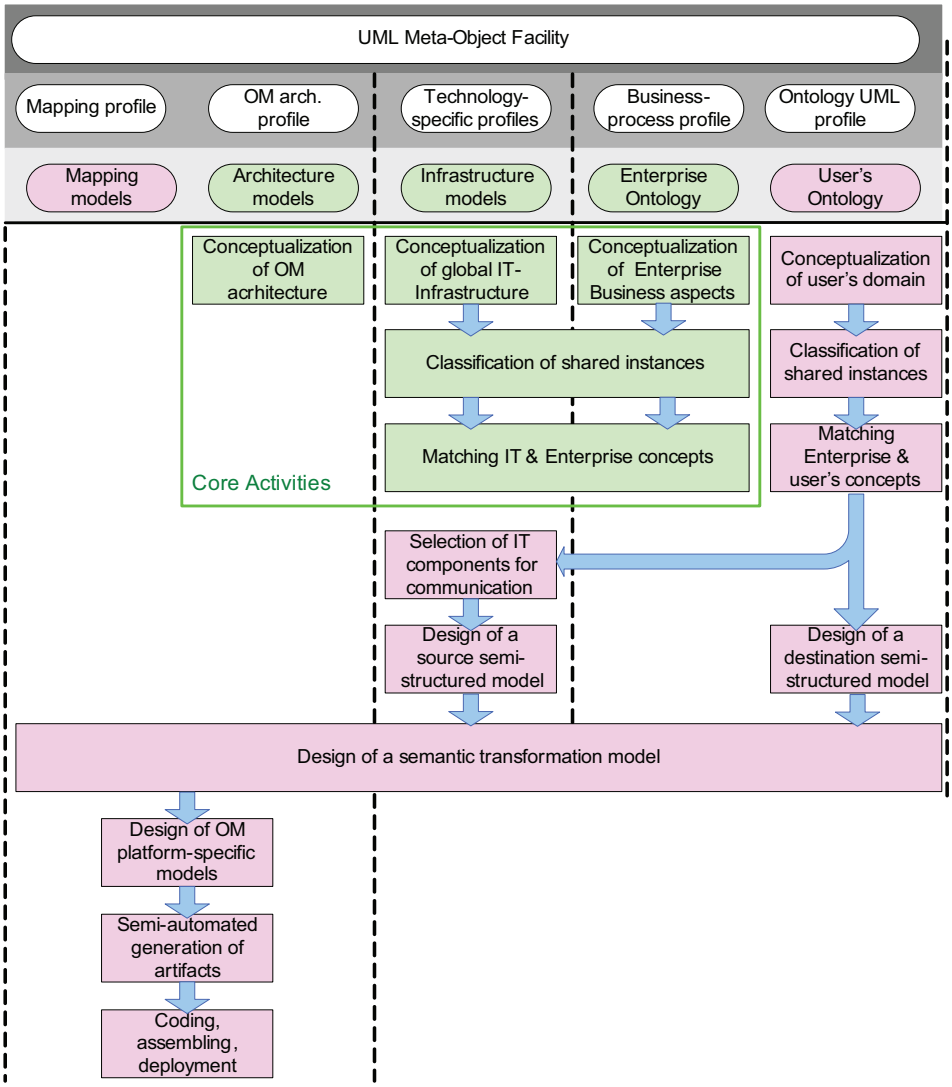

Fig. 2. Proposed methodology of Ontology Mediator's design and implementation. Dotted filling denotes initial core activities; dark colour defines repeatable activities during design of specific Ontology Mediator. UML models of the level 'M1' are the results of activities of the correspondent track

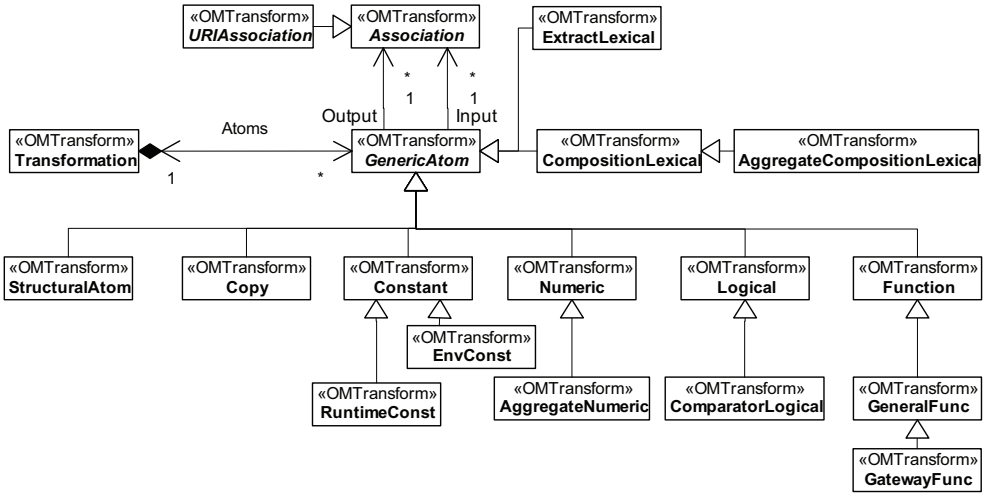

Fig. 3. The Transformation UML Profile

As soon as the users approve the content of the semantic transformation model it becomes a basis for design of platform specific models describing a concrete instantiation of Ontology Mediator. The platform-specific models determine which reusable hardware and software components will be combined together and which extension interfaces should be implemented during manual coding. Once the platform-dependent models have been produced the platform designers apply generation algorithms, which produce a skeleton of source code, as well as a list of additional hardware components and recommendations for selection of suitable base hardware modules. During final assembly of Ontology Mediator the programmers write small portions of glue code extending the generated skeleton, and the hardware engineers equip base hardware modules with additional components. This stage finishes process of Ontology Mediator development and the ready end-user product is shipped to the customer or is deployed in to the existing IT-infrastructure.

5. Framework architecture

According to the proposed methodology we developed a consistent ontology-based framework supporting design and development of Ontology Mediators. The framework has client-server architecture, and consists of front-end and back-end components promoting team work (fig.4).

The graphical front-end supports design of ontologies and UML-models for semantic transformations. Based on the Rational Software Architect platform by IBM and Eclipse Modelling Framework, the front-end adds several plugins, which implement a new visual modelling principle called Semantic Transformation Lasso (SETRAL), and provide interfaces to the framework back-end. SETRAL is extremely useful when modern Tablet PC and Visual Interactive Desks can be used during conceptualization and matching different ontology's concepts. For the user SETRAL offers a special smart graphical tool - Semantic Lasso (SL). That is a closed shape with arbitrary smooth boundaries. The user can draw a free-hand fragment of the SL boundary (the path) on top of the source UML class model to select certain classes to be matched. SETRAL automatically closes the path has been drawn

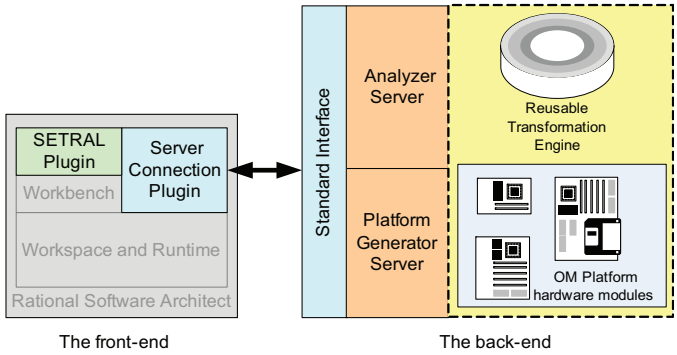

Fig. 4. Ontology-based framework for design and development of Ontology Mediators

and makes it smooth as necessary. Additionally SETRAL analyzes the source model and the meta-model to detect other entities (classes) that are semantically closely related to the entities inside the SL, but were not manually selected by the user. SETRAL proposes to include related entities with automatic extension of the SL's shape as needed. As soon as the source content of the SL was defined, SETRAL performs semantic mapping of the source entities inside the SL to the correspondent entities of the destination model. The result is displayed inside the SL as a fragment of the destination UML model. The level of transparency inside the SL can be adjusted interactively, so the user sees either two models or only one of them. Inside the SL the user can select entities of the destination model and switch to another editors to see the complete result of the matching. She/he has a possibility to easily return to the original editor with the active SL. SL tool can be used also for comprehensive analysis of the class instances. In this case after definition of the classes inside the SL, the user can run special SETRAL algorithms that will automatically generate instances with populated attributes.

The framework back-end contains the Analyzer Server and the Platform generator Server. The former server is closely related with the mapping design GUI. That component implements mathematical models of ontologies matching and models transformations as well as provides interfaces for loading core meta-models, domain models, domain-specific constraints in terms of OCL, as well as specifications of target software and the hardware platform. In accordance with loaded models the Analyzer Server automatically finds correspondent classes and attributes, returning results to the design GUI for further analysis by experts. The Platform Generator Server adopts basic principles of Model-Driven Architecture and automatically generates software artefacts for the selected target transformation platform. Each kind of the supported target platform uses the same library of software components (T-Engine). T-Engine contains algorithms for runtime semi-structured data transformation and business process enactment. Components of T-Engine can be deployed into the application server, the multi-agent system, or into the embedded platforms. In the later case our own specialized hardware platform is used.

6. T-Engine: architecture and implementation concepts

T-Engine is parted into four architectural layers, providing for different aspects of semantic interoperability: the External Communication Media layer, the RDF-mediation layer, the Internal Communication Media layer, and the Semantic Transformation layer (fig.5).

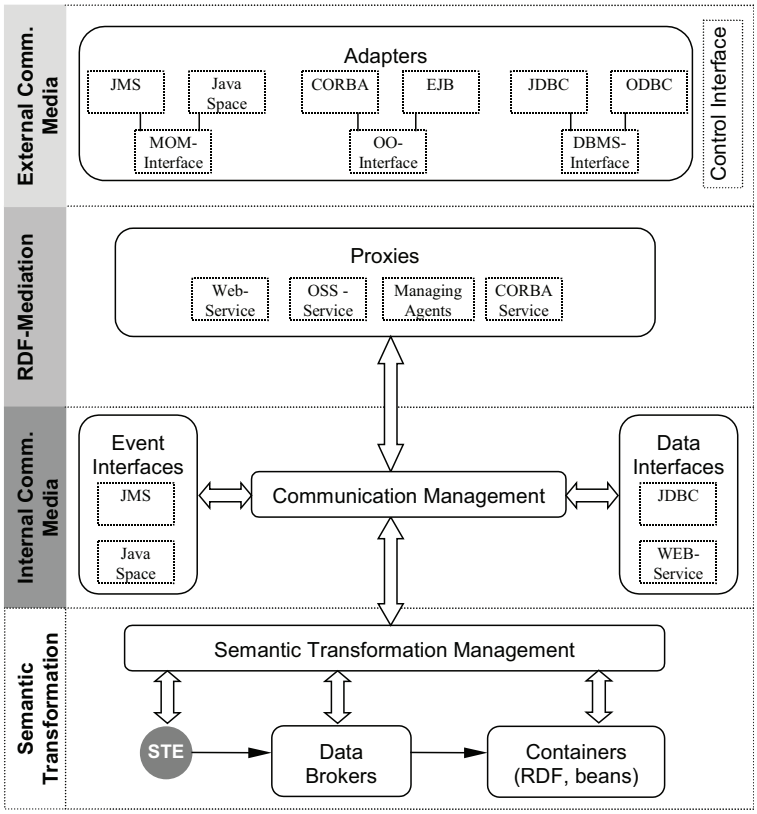

Fig. 5. Detailed architecture of T- Engine

The External Communication Media layer gives the implementation-neutral interface to outer distributed components of UCE. This layer comprises three unified operational interfaces to deal with major classes of modern distributed technologies (namely, Message-Oriented Middleware, remote calls of object's methods, Database Management Systems), as well as so-called adapters. Each adapter implements certain operational interface by means of concrete libraries and vendor-specific tools. A separate part implements control and management interface of T-Engine.

The RDF-Mediation layer provides for encapsulation of heterogeneous data structures and communication algorithms in accordance with a single message-oriented paradigm. Following this paradigm the correspondence is set between every communication act with an external component and a message of the certain type. Interacting with external components the RDF-Mediation Layer creates one or many message instances, and passes them to the Internal Communication media Layer for further use. Similar to the adapters, at the RDF-Mediation Layer a special component called Proxy takes responsibility of messages' production for a certain technology.

The produced message consists of actual data embodied in the form of RDF, and internal meta-information (e.g. type qualification, message's timestamp or a message's producer). The type of the message poses restrictions on its possible data structure. These restrictions

are expressed in the form of the RDF schemata and define allowable atomic fields and sub-structures for any valid message instance of this type. Thus the message type can be concerned likely a micro-ontology, and it can be created and modified in the framework front-end during designing the source semi-structured model. Generalization of information interchange in the form of message's flows promotes uniform representation of UCE component's behaviour and data inside the underlying layers: components play roles of message consumers or producers and their interaction can be represented as modification, creation or transformation of message's instances.

The Internal Communication Media layer supports asynchronous message-based communication between the layers and provides for transaction management. Interacting with the Communication Management module, different modules of T-Engine subscribe to messages of particular type. Being notified of occurrence of new messages, the modules-subscribers fetch the messages from the internal queue, transform them, and put newly produced messages back for shared use.

To improve performance and reduce the message-processing costs the Communication Management module also promotes separation of event and data flows as much as possible avoiding passing large RDF models inside the message body at the intermediate stages of the message processing. Once submitted to the Communication Management module, the message is analyzed and rearranged to store either the original RDF model or only short pragmatic instructions. In later case the module of Data Interfaces uses pragmatic instructions for the "lazy" information retrieval from external data sources in order to reconstruct the complete RDF model. In the module of Event Interfaces all intermediate message processing tasks such as routing, filtering, collecting are performed on the basis of the internal meta-information without touching upon the correspondent RDF model. Hence the creating of the content of the huge RDF model can be postponed up to the moment of its actual use, but in the case of the small RDF model its content is completely stored inside the message.

The Semantic Transformation layer plays the central role in the successful fulfilment of Ontology Mediator's activities. In particular, on this layer the Semantic Transformation Management module collects instances of input messages and fuses them into the united RDF model, which is in direct correspondence with the source semi-structured model produced as the result of our design methodology. Relating message's instances with each other, the Semantic Transformation module exploits the message's meta-information and capabilities of RDF framework to link different resources via universal resource identifiers (URI). Once all needed instances have been collected and the source RDF model has been completely composed in accordance with the specification, the Semantic Transformation Management module places the model into the container for further processing by means of so-called Semantic Transformation Entities (STE).

Each STE serves as an independent workflow process reacting on completed model's appearance in the particular container. T-Engine allows dynamic independent deployment of different STEs and later manages their concurrent execution. STEs can be organized into chains of linked semantic transformations establishing complex information processing inside Ontology Mediator. Through the interface of DataBrokers a single STE gains access to the elements of the united RDF model, and performs transformation of this model to the destination RDF model in accordance with the semantic transformation UML model. When the STE successfully finishes building of the destination RDF model, algorithms of the Semantic Transformation Management module split this model into separate RDF sub-models corresponding to distinct messages. Carrying on the information in terms of the

user's ontology, messages proceed to the External Communication Media layer. Alternatively, the messages can take part in construction another source RDF model.
The architecture of T-Engine, as it has been described in previous paragraphs, is almost technology independent. Indeed, different modern distributed technologies such as CORBA, EJB, JMS and JINI can be used for practical implementation. In the course of various software prototypes design we have found that although CORBA and EJB are more widely used, the JINI-based implementation gives many attractive features. We have found that JINI technology is the most suitable for implementation of proxies on the RDF-Mediation layer. In this case JINI Smart Proxies can be installed quickly for preparation of the RDF-model. Smart JINI proxies expose a unified interface to Event Interfaces, Control Interfaces and Data Interfaces. These proxies are available via JINI lookup service (reggie). CORBA and JMS implementations of Event Interfaces were also performed but they are not described here.

7. The extensible hardware platform for ontology mediator

Our investigations of several use cases determined major guidelines for hardware architecture of Ontology Mediators. It should have nearly the same customer properties as sensor network devices: reasonably low prices, friendly interface for installation and maintenance. These properties allow for preserving unique features of sensors networks and deploy Ontology Mediators in different locations following changing and diverse needs of the end-users. At the same time Ontology Mediator should provide for a wide range of different mediation scenarios, and, in our vision, its hardware should be principally able to support communication via any of the most popular wireless protocols as well as wired ones (Ethernet, RS232/435, CAN). Later requirement leads to broad variations in internal hardware and software architecture.
Fitting the proposed model-driven design methodology the extensible hardware platform comprises two specially designed hardware modules:

- Processing Unit – an unmodified part of Ontology Mediator, where the central processor and basic communication interfaces are installed.
- Extension Board – a customizable interface board that can be easily extended by different hardware and embedded software components (plug-ins) on demand of specific requirements.

A number of design solutions were studied and prototypes were developed in order to find the most suitable decomposition of functions and balanced costs. Our latest experimental implementation of Ontology Mediator's Processing Unit has following features (Fig.6-a):

- MCU: LPC2294 16/32 bit ARM7TDMI-S™t with 256K Bytes Program Flash, 16K Bytes RAM, 4x 10 bit ADC, 2x UARTs, 4x CAN, I2C, SPI, up to 60MHz operation.
- 4MB SRAM 4x K6X8008T2B-F/Q SAMSUNG.
- 4MB TE28F320C3BD90 C3 INTEL FLASH.
- 10Mb TP Ethernet (CS8900A).
- OS: RTOS ECOS 2.0.
- Connectors: Power supply, RJ45, HDR26F (for connection to the extension board).

For the developed Processing Unit a reduced version of the Ontology Transformation Engine was developed. In that version transformation algorithms, internal control and interfacing functions were programmed in C and C++ programming languages. Now we are at the final stage of porting Wonka Java virtual machine (Wonka, 2010) to Processing

Unit hardware platform that will allow enriching application capabilities and provide full-fledged implementation of the Engine.

(a) (b)

Fig. 6. Hardware components of the extensible platform: a- Processing Unit in the case of stand-alone usage without Extension Board; b – an experimental sensor

Fig. 7. Specialization of Ontology Mediator for telecommunication domains

During experiments with Ontology Mediators we paid attention to semantic transformation algorithms, and used multiple wired sensors to reduce implementation costs. So, now Extension Board has support of both wired and wireless protocols: I2C, two CAN, 8x30 sensors (Fig.6-b), SPI-to-UART MAX3000 transceiver for connection to wireless SANET via radio transceiver. We place to our nearest plans generalization of the board architecture and its redesigning in the framework of evolvable hardware paradigm. The final version of Extension Board will include FPGA-based evolvable hardware part and a set of free slots for drivers and transceivers.

Most of the experiments with ontology mediation algorithms employed two wired SANETs, directly connected to Ontology Mediator via RS485 interface. The first test-bed SANET with ring topology consists of 70 sensors and actuators deployed to a rail road model. Ontology

Mediator presents dynamic state of the rail road in terms of an operator and allows performing basic actions by friendly Web-based interface (Java applets). The second SANET has star topology with eight rays. Each ray is capable of supporting up to thirty sensors. This network was installed on a real test site to monitor environmental conditions. In that case, Ontology Mediator (Fig.7) provides operators with real-time information about operational conditions, as well as equipment performance degradation forecasts. For these specific purposes, specialization of Ontology Mediator includes equipping Extension Board with LCD monitor and controls buttons, developing models and ontology mapping between low-level measurements of temperature, humidity and gas contamination to high-level abstractions of equipment operators to integrate SANET with external information systems.

8. Conclusion

In this work we attracted attention to achieving semantic interoperability in the context of wide-area Ubiquitous Computing Environment. The concept of "Ontology Mediator" was offered to designate a specialized device or a software component, which goal is fusion of sensors data, their smart transformation and expression in various forms of real-world concepts by application of ontologies manipulation. The developed design methodology and supporting framework provide foundations for model-driven development of Ontology Mediators. The basis of our approach is the ontology transformation process with locally defined conditions for joining separate input messages into the united RDF model and explicitly defined rules of its transformation into the set of output RDF models. Implementation of that process was done in the library of reusable software components (T-Engine). The T-Engine's components implement original message-oriented algorithms for semantic consistency testing, consolidation and transformation. During software implementation of T-Engine different distributed technologies were exploited and the tuple space based communication paradigm based on JavaSpaces technology showed the best performance and was the most attractive one from the architecture point of view. To study effects and benefits of embedded T-Engine's implementations we developed an extensible specialized hardware platform.

In comparison with known middleware architecture approaches for sensors networks [6, 13] our solution operates on a level of meta-data and allows bridging a gap between different information object-oriented models. In fact, having interfaces to different communication protocols, Ontology Mediator can be applied for orchestrating heterogeneous middleware services when the environment consists of different sensors networks and enterprise-wide applications. Increased reusability is seemed to be another attractive feature of our approach. During the Ontology Mediator's design, the platform designer accumulates knowledge about certain application domains and patterns of behaviour for different customers and needs. It allows continuous extending library of available models; moreover third party developers and teams can share such library. At the same time application of the single meta-model for development of different models allows simplify models transformation and integration.

Despite of using the same general ontology paradigm for meta data representation proposed in (Chen et al., 2004), we contribute original design methodology and the software-hardware framework that allow implementing solutions of various scales, performing ontology transformation simulations and testing.

The generic layered approach to design allows easy adaptation of the framework for different distributed systems. Regardless of components structure's modification and evolutionary variations in semantics, the developed solution allows for maintaining the permanent information consistence among multiple components. For example in a case of modification of the underlying database schema or the interface definition it is necessary only to modify the definition of semantic transformation for some STE without touching upon the implementation of external components.

Solutions like TSpaces (IBM), MARS-X (Cabri et al., 2001) and XMIDDLE (Mascolo et al., 2001) properly illustrate trends in application of modern Internet technologies such as XML and RDF in coordination frameworks. In our opinion MARS-X has some common points of interest and can be compared with our framework. Both approaches use tuple based coordination paradigm based on JavaSpaces software implementation. In MARS-X tuples are used for the XML document storing and different software agents can extract relevant information using complex search patterns defined programmatically. In our approach tuples are used to notify about the message instance presence and components define conditions for joining messages on the basis of its properties by a declarative way. As for semantic interoperability the important restriction of MARS-X is necessity to use the same structure of XML document (share the same DTD) for all components of a distributed system. Modification of DTD will require redesign of software components that is not always possible in the loosely coupled system. In our case the declarative description of joining conditions and delegating messages collecting and transformation activities into the separate middleware service allows to separate design of component algorithms from management of semantic interoperability.

Considering implementation issues of our framework it can be mentioned that the designed software architecture allows for rapid inclusion of new software technologies on different levels without significant changes of the core. Splitting the process of the message processing from the process of huge RDF models retrieving makes Ontology Mediator more robust.

9. References

Abiteboul, S.; Hull, R. & Vianu, V. (1995). *Foundations of databases*, Addison Wesley Publ. Co., Reading, ISBN 0-201-53771-0, Massachussetts

Abowd, G.D.; Mynatt, E.D. (2000). Charting Past, Present, and Future Research in Ubiquitous Computing,. *ACM Transaction on Computer-Human Interaction*, Vol. 7, No. 1, pp. 29-58, ISSN 1073-0516

Ahamed, S.I.; Vyas, A. & Zulkernine, M. (2004). Towards Developing Sensor Networks Monitoring as a Middleware Service, *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW '04)*, pp. 465-471, ISSN 1530-2016, Montréal, Québec, Canada, August 15-18, 2004

Barwise, J.; Seligman, J. (1997). *Information Flow*, Cambridge University Press, ISBN 0-521-58386-1, Cambridge

Branch, J.W. ; Davis, J. ; Sow, D. & Bisdikian, C. (2005) Sentire: A Framework for Building Middleware for Sensor and Actuator Networks, *Proceedings of the 1st IEEE International Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS'05)*, pp. 396-400, Kauai Island, Hawaii, March 8-12, 2005

Buneman, P. (1997). Semistructured data, *Proceedings of the ACM Symposium on Principles of Database Systems*, pp.117-121 , ISBN 0-89791-910-6, Tucson, Arizona, United States, 1997, ACM, Tucson

Buneman, P.; Davidson, S. ; Fernandez, M. & Suciu, D. (1996). Adding structure to unstructured data, *Technical Report MS-CIS-96-21,* University of Pennsylvania, Computer and Information Science Department

Cabri, G. ; Leonardi, L. & Zambonelli, F. (2000). XML dataspaces for mobile agent coordination.", *Proceedings of the 2000 ACM symposium on Applied computing*, pp. 181-188, ISBN 1-58113-240-9, 2000

Chan, E., Bresler, J. ; Al-Muhtadi, J. & Campbell, R. (2005). Gaia Microserver:An Extendable Mobile Middleware Platform, *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005)*, pp. 309-313, Kauai Island, Hawaii, March 8-12, 2005

Chen, H.; Perich, F.; Finin, T. & Joshi, A. (2004). SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications, *Proceedings of the 1st Annual International Conference on Mobile and Ubiquitous Systems:Networking and Services (MobiQuitous '04)* , pp. 258-267, ISSN 0-7695-2208-4, Boston, Massachusetts, August 22-25, 2004

Crossbow Technology,Inc., In http://www.xbow.com.

Curino, C.; Giani, M. ; Giorgetta, M. ; Giusti, A.; Murphy, A.L. & Picco, G.P. (2005). Tiny LIME :Bridging Mobile and Sensor Networks through Middleware, *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005)*, pp. 61-72, Kauai Island, Hawaii, March 8-12, 2005

Dragan, G.; Dragan, D. & Vladan D. (2006). *Model Driven Architecture and Ontology Development*, Springer-Verlag, ISBN: 978-3-540-32180-4

Ehrig, M. (2007). *Ontology Alignment : Bridging the Semantic Gap. Series: Semantic Web and Beyond, Vol. 4*. Springer-Verlag, ISBN 978-0-387-32805-8)

Feng, J.; Koushanfar, F. & Potkonjak, M. (2002). System-Architectures for Sensor Networks Issues, Alternatives, and Directions, *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'02)*, pp. 226, ISSN 1063-6404, Rio de Janeiro, Brazil, September 16-18, 2002

Gruber, T.R. (1995). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human-Computer Studies*, No. 43(5/6), 1995, 907-928, ISSN 1071-5819

Herring, C. (2000). Microprocessors, Microcontrollers, and Systems in the New Millennium, *IEEE Micro*, Vol. 20, No. 6, Nov/Dec, 2000, 45-51, ISSN 0272-1732.

Hoffmann, B. & M. Minas, M. (2000). Towards Generic Rule-Based Visual Programming, *Proceedings of the 2000 IEEE International Symposium on Visual Languages (VL'00)*, pp. 65-67, ISBN 0-7695-0840-5, Seattle, Washington, 2000

Hönle, N.; Käppeler, U.-P.; Nicklas, D.; Schwarz, T. & Grossmann, M. (2005). Benefits of Integrating Meta Data into a Context Model, *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005)*, pp. 25-29, Kauai Island, Hawaii, March 8-12, 2005

Jeng, T. (2004). Designing a Ubiquitous Smart Space of the Future: The Principle of Mapping, *Proceedings of International Conference of Cognition and Computation (DCC '04)*, ISBN 978-1-4020-2392-7, MIT, Cambridge, 19-21 July, 2004

Kalfoglou, Y. (2001) Exploring ontologies, In : *Handbook of Software Engineering and Knowledge Engineering*, *Vol. 1, Fundamentals*, S.K. Chang (Ed.), 863–887, World Scientific, ISBN 978-9810249731, Singapore

Kawahara, Y. ; Minami, M.; Saruwatari, S.; Morikawa, H. & Aoyama, T. (2004). Challenges and Lessons Learned in Building a Practical Smart Space, *Proceedings of the 1st Annual International Conference on Mobile and Ubiquitous Systems:Networking and Services (MobiQuitous '04)*, pp. 213-222, ISBN 0-7695-2208-4, Boston, Massachusetts, August 22-25, 2004

Mascolo C. ; Capra, L. & Emmerich, W. (2001) An XML-based Middleware for Peer-to-Peer computing, *Proceedings of the the First Int. Conf. on Peer-to-Peer Computing (P2P'01)*, pp. 69-74, ISBN 0-7695-1503-7, Lingkoping, Sweden, 2001

Misc.Tinyos: A component-based os for the networked sensor regime. In http://webs.cs.berkeley.edu/tos/

Niemelä, E. ; Latvakoski, J. (2004). Survey of requirements and solutions for ubiquitous software, *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pp. 71-78, ISBN 1-58113-981-0, College Park, Maryland, 2004

Oliver, I. (2005) Applying UML and MDA to Real Systems Design, *Proceedings of the conference on Design, Automation and Test in Europe (DATE'05)*, Vol.1, pp.70-71, ISSN 0-7695-2288-2, Munich, Germany, 7-11 March, 2005

Rozenberg, G. (ed.) (1997). *Handbook of Graph Grammars and Computing by Graph Transformations", Volume 1: Foundations*. World Scientic, ISBN 981-02-2884

Sowa, J.F. (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks Cole, ISBN 0-534-94965-7, Pacific Grove, CA

Tilak, S.; Abu-Ghazaleh, N. & Heinzelman W. (2002). A Taxonomy of Wireless Micro-Sensor Network Models, *ACM Mobile Computing and Communications Review*, Vol. 6, No. 2, 2002, 28-36, ISSN 0146-4833

Tokunaga, E.; Zee, A. van der; Kurahashi, M.; Nemoto, M.& Nakajima, T. (2004). A Middleware Infrastracture for Building Mixed Reality Applications in Ubiquitous Computing Environments, *Proceedings of the 1st Annual International Conference on Mobile and Ubiquitous Systems:Networking and Services (MobiQuitous '04)*, pp. 382-391, ISSN 0-7695-2208-4, Boston, Massachusetts, August 22-25, 2004

Tsetsos, V.; Alyfantis, G. ; Hasiotis, T. ; Sekkas, O. & Hadjiefthymiades S. (2005). Commercial Wireless Sensor Networks: Technical and Business Issues, *Proceedings of the Second Annual Conference on Wireless On-demand Network Systems and Services (WONS'05)*, pp. 166-173, ISSN 0045-7906, St. Moritz, Switzerland, January 2005

Volgyesi, P.; Ledeczi, A. (2002). Component-Based Development of Networked Embedded Applications, *Proceedings of the 28th Euromicro Conference (EUROMICRO'02)*, pp. 68, ISSN, Dortmund, Germany, 4-6 September, 2002

Wonka Java VM Project. In http://en.wikipedia.org/wiki/Wonka_VM. 2010

Zhou, Y. ; Zhao, Q. & Perry M. (2005). Reasoning over Ontologies of On Demand Service, *Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05)*, pp. 381-384, ISSN 0-7695-2274-2, Hong Kong, 29 March-1 April, 2005