Towards Semantic-Supported SmartLife System Architectures for Big Data Services in the Cloud

Eman El-Sheikh
University of West Florida
Department of Computer Science
Pensacola, FL USA
eelsheikh@uwf.edu

Sikha Bagui
University of West Florida
Department of Computer Science
Pensacola, FL USA
bagui@uwf.edu

Donald G. Firesmith
Carnegie Mellon University
Software Engineering Institute
Pittsburgh, PA USA
dgf@sei.cmu.edu

Ilia Petrov
Reutlingen University
Data Management Lab
Reutlingen, Germany
ilia.petrov@reutlingen-university.de

Norman Wilde
University of West Florida
Department of Computer Science
Pensacola, FL USA
nwilde@uwf.edu

Alfred Zimmermann
Reutlingen University
Architecture Reference Lab
Reutlingen, Germany
alfred.zimmermann@reutlingen-university.de

***Abstract* – SmartLife applications are emerging as intelligent user-centered systems that will shape future trends in technology and communication. The development of such applications integrates web services, cloud computing, and big data management, among other frameworks and methods. Our paper reports on new perspectives of services and cloud computing architectures for the challenging domain of SmartLife applications. In this research, we explore SmartLife applications in the context of semantic-supported systems architectures and big data in cloud settings. Using a SmartLife application scenario, we investigate graph data management, fast big data, and semantic support through ontological modeling. The ontological model and architecture reference model can be used to support semantic analysis and program comprehension of SmartLife applications.**

Keywords – SmartLife applications; Semantics and Ontology; Big Data Management; Enterprise Systems Architecture; Services–Oriented Architectures; Cloud Computing.

I. INTRODUCTION

Information and data are central components of our everyday activities. Social networks, smart portable devices, and intelligent cars, represent a few instances of a pervasive, information-driven vision we call SmartLife. Imagine speeding on the motorway, and receiving a text message on your mobile device from a friend asking to meet you. Since the number is on your personal contact list, the message is then transferred to your car's personal information system and read as well as displayed as soon as the traffic conditions allow that. You accept the invitation and the system checks recent social postings of your friend to recommend possible locations. Your friend has 'liked' multiple Espresso postings lately so the system infers that he/she might enjoy having one and executes a query for excellent Espresso places nearby and for times convenient for both your schedules. The SmartLife system would use your social profiles to recommend places. It would verify which of your close friends you have not met for a while are available and what they might have recommended, compile a simple list and display it on your car's head-up display. If you agree it will schedule an appointment in all personal calendars, distribute routes and queue them in the navigation systems, possibly recommending parking places. Your status messages will be automatically updated. Of course, you get to pick the best coffee blend and roast yourself!

The above is an example of a service-based semantically rich application scenario. Social graph analysis and management, big data, and cloud data management are essential to the above scenario. Ontological analysis, smart devices, personal information systems, hard non-functional requirements, such as location-independent response times and privacy, are some of the basic concepts in building such a SmartLife scenario.

Additional application domains of the SmartLife vision include: (i) intelligent mobility systems and services; (ii) intelligent energy support systems; (iii) smart personal health-care systems and services; (iv) intelligent transportation and logistics services; (v) smart environmental systems and services; (vi) intelligent systems and software engineering; (vii) intelligent engineering and manufacturing.

A. Research Questions

This paper describes work in progress that will address the following research questions:

1. How can service-oriented architectures (SOA) and enterprise systems architectures support SmartLife applications?
2. Is an ontological modeling approach useful to support semantic SmartLife applications?
3. How can semantic modeling approaches be effectively combined with system application

engineering, big data and services computing in the cloud?
4. How are semantic and social data in the SmartLife scenario efficiently managed as big data? How can real-time analysis and updates be efficiently performed on big graph data in cloud settings?
5. What are suitable approaches to enterprise IT-architectures for services and cloud computing, guiding the management and control of SmartLife scenarios?
6. How should existing software engineering methods evolve to cover SmartLife services?

B. *Impact*

The technological and business impact of the SmartLife vision has multiple aspects. While the business side targets intelligent approaches and structural business, the technological side is more diverse. Expected fields of innovation include:

- Software engineering methods for cloud applications
- Influence of dynamic configuration components for products, processes and systems
- Graph databases for fast big data and new hardware technologies
- Advanced architectural approaches for reconfigurable pervasive and mobile scenarios based on service-oriented and cloud computing architectures
- Common sematic approaches as a basis for modeling smart application scenarios for user-centered systems.

The rest of the paper describes the framework and methodology for the proposed research. Section II describes a minimalistic configuration scenario for SmartLife. Section III describes the implications and research issues resulting from SmartLife data management focusing on Big Graph Data Management. Section IV targets semantic representations and mechanisms for intelligent SmartLife support. Section V integrates both business and computer science aspects of a consistent configuration of enterprise systems architecture, and section VI concludes the paper.

II. SMARTLIFE APPLICATION SCENARIOS

SmartLife applications span a broad range of domains including intelligent configuration services, intelligent transportation and logistics services, personal health care systems and services, smart environmental systems, and intelligent engineering and manufacturing systems. Below we describe a simple starting scenario for SmartLife.

WebAutoParts is a hypothetical online automobile parts dealer intended to model an Internet start-up company that is using SOA for rapid development [1]. Its software uses BPEL for orchestration of commercially available external services from well-known vendors. As shown in Figure 1, the Order Processing workflow for WebAutoParts has two stubbed in-house BPEL services (OrderProcessing and InventoryRepository) and four commercially-available

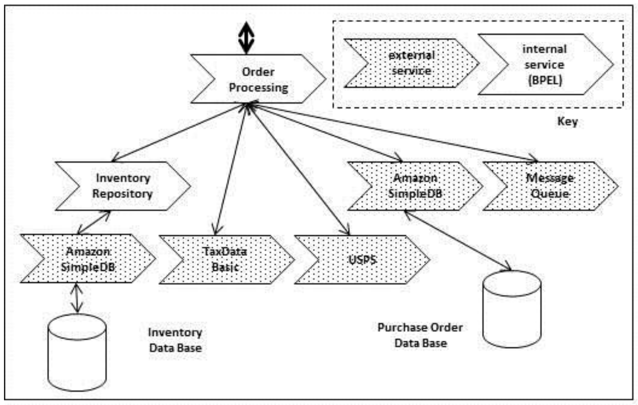

Figure 1. WebAutoParts: Services in the Order Processing Workflow

external services: Amazon Web Services - *SimpleDB* (data base) and *SimpleQueueService* (message queuing); StrikeIron.com - *TaxDataBasic* (sales tax rates); Ecocoma - *USPS* (shipping costs).

WebAutoParts is much smaller than most real SOA applications. However, it is useful for ontological exploration since it consists of syntactically correct BPEL code and contains XSD and WSDL documents typical of current industrial practice. We will also explore the use of other SOA systems for SmartLife domains, such as intelligent transportation services and intelligent engineering and manufacturing systems in future research work.

III. GRAPH DATA MANAGEMENT

In terms of data management, SmartLife systems manage and analyze big data. Major components are social, enterprise, semantic, and sensor data [2]. We actively investigate the following research aspects:

- Heterogeneity. Data from multiple possibly heterogeneous data sources have to be federated. New approaches to data fusion and cleaning are needed.
- High Volume. Data is being produced at high rates on the scale of Petabytes or Exabytes from many users and data sources. In contrast to traditional data management it is not reasonable to assume upper bounds. It is mostly distributed.
- High Update Rates. Data and content are being produced at high rates for new activity (tweets, social graph updates and social content, sensor and mobile data). Hence a paradigm shift is required in big data management and high update rates.
- Near/Real-time Analytics. Near-time analytics and discovery are a prerequisite for successful SmartLife systems. In addition to traditional analytics, data mining and information retrieval, SmartLife systems are expected to offer user recommendations. The near-time character of data-analysis requires new approaches.

A research area that gains significant attention and offers a common way of data processing and analysis is graph data management. Existing approaches, algorithms and systems need to be reevaluated in the above context [3]. The use of novel hardware such as many-core CPUs, FPGAs, new storage technologies, like Non-Violate and Flash memories, are critical to handle the high update rates and near-time analytics.

One research goal is to investigate graph database systems [4] in a cloud setting that handle huge data volumes and high update rates and at the same time offer near-time analytics, recommender functionality and crowdsourcing. The efficient use of new hardware technologies is another key research goal.

Social big data imply processing of very large graphs that are typically maintained at multiple sites (cloud settings) with high update rates [5]. Traditionally high volume graph data is being handled by disc-based graph databases, which are too slow to handle the complexity of the typical inference and analytics graph queries. Low response times represent a key non-functional requirement. Additional performance related research issues arise from the need to handle mixed loads – complex graph analytics as well as high update rates. The efficient use of new hardware is a key requirement to meet these performance challenges, which translates into a number of research issues: (i) optimal use of flash and non-volatile memories since many of the current algorithms are not suitable; (ii) efficient use of multi-core CPUs and FPGAs for graph data analysis; and (iii) distribution and synchronization problems in Cloud settings.

IV. Semantic Support Through Ontologies

A. *Development of an Ontological Model*

The Open Group developed and released the SOA Ontology 2.0 [6]. This ontology has two main purposes:

1. It defines the concepts, terminology and semantics of SOA in both business and technical terms.
2. It contributes to model-driven SOA implementation.

The Open Group's SOA Ontology [6] is represented in the Web Ontology Language. The Open Group ontology contains classes and properties corresponding to the core concepts of SOA. The formal OWL definitions are represented (i) in the OWL syntax; (ii) as UML models of the concepts and their relationships; and (iii) all models are supplemented by natural language descriptions.

B. *Ontological Model for SmartLife Applications*

The phase after development of the ontological model will focus on exploring how the model can be used to support program comprehension for SOA-based SmartLife systems. Several specific SOA comprehension tasks will be identified, including (but not limited to):

- Impact analysis (If X is changed, what additional changes may be needed?)
- Concept location (Where is concept Y implemented in this system?)

We will explore visualization of the ontological model developed to support: (i) system comprehension and (ii) information and data management. Additional research questions include: Is the ontological reference model sufficient to model a SOA SmartLife system? Are there gaps? Would the reference model need to be extended?

C. *Comparing the Ontological Approach to Other Knowledge Modeling Approaches*

1) Concept Maps

Concept maps are an established framework for organizing and representing knowledge [7]. A concept map is a diagram that shows the relationships among concepts. Concepts, usually represented as boxes or circles, are connected with labeled arrows in a downward-branching hierarchical structure. The relationship between concepts can be expressed in linking phrases such as "is" or "includes." Concept maps are particularly useful for analyzing and organizing large and complex domains. Concept maps can be structured hierarchically, linked together, and augmented with other resources such as text, graphics, videos, etc., to create a knowledge model [8].

2) Entity Relationship Model

The Entity Relationship (ER) model is an established data modeling technique, well accepted in the database world [9]. The ER model is used to visually represent data in databases in terms of entities, their attributes and relationships. Entities describe a complex structured concept like a person, place, thing or event of interest. Attributes are used to describe entities. Attributes can be either single value or multi-valued. And, relationships describe associations among entities. Relationships are explained in terms of their connectivity (or cardinality), and their connectivity can be indicated by one-to-one (1:1), one-to-many (1:M) and many-to-many (M:N) relationships. Cardinality is related to upper and lower bounds. Participation in this connectivity by member entities may be optional (partial) or mandatory (full).

3) Unified Modeling Language Model

Unified Modeling Language (UML) is a standardized general-purpose modeling language used in object-oriented software engineering. The standard is managed, and was created, by the Object Management Group.

UML is used to specify and visualize the artifacts of an object-oriented software-intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as: activities, actors, business processes, database schemas, and (logical) components, programming language statements, reusable software components. UML combines techniques from data modeling (ER modeling), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies.

4) Tree Abstractions

The tree representation is a hierarchical representation of the data, mainly used in the XML data format [10]. This structure allows representing information using parent/child relationships: each parent can have many children, but each child has only one parent (also known as a 1-to-many relationship). All attributes of a specific record are listed under an entity type.

We are exploring the use of concept maps and other knowledge models for semantic analysis of SmartLife applications. The ontological model developed for the target applications can be compared to a knowledge model of the application to identify similarities and differences between the two program comprehension approaches, as well as strengths and weaknesses of each approach.

V. ENTERPRISE SYSTEMS ARCHITECTURE

In areas where flexibility or agility in business is important, services computing is the approach of choice to organize and utilize distributed capabilities. Innovation oriented companies have introduced in recent years service-oriented architectures to assist in closing the business - IT gap and making it cloud-ready. The benefits of SOA are recognized for systems on the way to cloud computing and being ready for extended service models. They comprise flexibility, process orientation, time-to-market, and innovation.

A. *Reference Architectures for Services & Cloud Computing*

The OASIS Reference Model for Service Oriented Architecture [11] is an abstract framework, which guides reference architectures [12]. The ESARC – Enterprise Services Architecture Reference Cube [13] (Figure 2) is more specific and completes these architectural standards in the context of EAM – Enterprise Architecture Management, and extends these architecture standards for services and cloud computing.

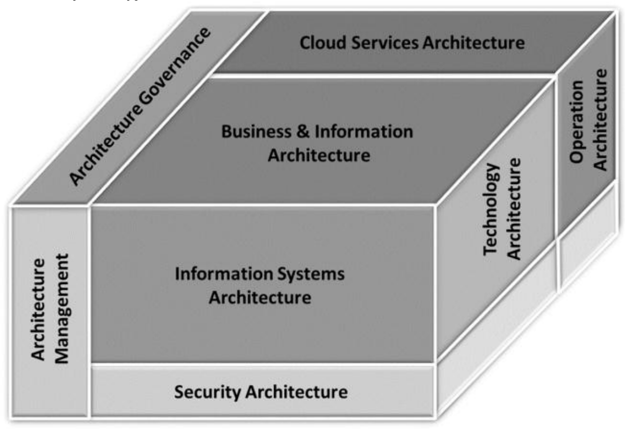

Figure 2. ESARC - Enterprise Software Architecture Reference Cube

ESARC provides an abstract model for application architectures and implementation of service-based enterprise systems. ESARC is an original architecture reference model, which provides an integral view for main interweaved architecture types. ESARC abstracts from a concrete business scenario or technologies. The Open Group Architecture Framework provides the basic blueprint and structure for our extended service-oriented enterprise software architecture domains like: Architecture Governance, Architecture Management, Business and Information Architecture, Information Systems Architecture, Technology Architecture, Operation Architecture, and Cloud Services Architecture. ESARC provides a coherent aid for examination, comparison, classification, quality evaluation and optimization of architectures.

The Business and Information Reference Architecture - BIRA (Figure 2) provides, for instance, a single source and comprehensive repository of knowledge from which concrete corporate initiatives will evolve and link. This knowledge is model-based and defines an integrated enterprise business model, which includes organization models and business processes. The BIRA opens a connection to IT infrastructures, IT systems, and software as well as security architectures. The BIRA confers the basis for business-IT alignment and therefore models the business and information strategy, the organization, and main business demands as well as requirements for information systems, such as key business processes, business rules, business products, services, and related business control information.

The ESARC Information Systems Reference Architecture –ISRA (Figure 2) is the application reference architecture and contains the main application-specific service types, defining their relationship by a layer model of building services. The core functionality of domain services is linked with the application interaction capabilities and with the business processes of the customer organization. In our research we are integrating the reference models for services computing [13].

Cloud architectures are still in development and have not yet reached their full potential of integrating EAM with Services Computing and Cloud Computing. Integrating and exploring these three architectural dimensions into consistent reference architectures is a basic part of our current research. The ESARC – Cloud Services Architecture (Figure 2) provides a reference-model-based synthesis of current standards and reference architectures, like [14].

B. *Architecture Metamodel and Ontology*

Metamodels are used to define architecture model elements and their relationships within ESARC. We use metamodels as an abstraction for architectural elements and relate them to architecture ontologies [15]. The OASIS Reference Model for SOA [11] is an abstract framework, which defines generic elements and their relationships for service-oriented architectures. This reference model is not a standard, but provides a common semantic model for different specialized implementations.

Reference architectures [12] are derived from a reference model. It is a composition of related architectural elements, which are built from typed building blocks as the result of a pattern-based mapping of reference models to software elements. Architecture patterns, as in [17], [18] are human readable abstractions for known architecture quality attributes, and represent standardized solutions, considering architectural constraints for certain recurring problems.

Architecture ontologies represent a common vocabulary for enterprise architects who need to share their information based on explicitly defined concepts. Ontologies include the ability to automatically infer transitive knowledge. The technical standard of service-oriented architecture ontology from [6] defines core concepts, terminology, and semantics of a service-oriented architecture in order to improve the alignment between the business and IT communities. The following stakeholders are potential users of the SOA ontology, related architecture metamodels, as well as concrete architectural building blocks: business people and business architects, information systems and software architects, architects for the technological infrastructure, cloud services architects and security architects. The metamodel of BIRA consists of ESARC-specific concepts, which are derived as specializations from generic concepts such as Element and Composition from the Open Group's SOA Ontology [6].

Using the ESARC ontology, we can navigate in the multidimensional space of enterprise architecture management structures and enable in a future research effort of semantic-supported navigation for architects as well as intelligent inferences. Additionally we want to add visualizations for these ontology concepts, as part of a sematic-supported architecture management cockpit.

C. *Methodology Framework for System Architectures*

As its name implies, the Method Framework for Engineering System Architectures (MFESA) [16] is a framework for using situational method engineering to create appropriate methods for engineering system architectures. MFESA consists of:

- An ontology that defines the concepts underlying system architecture engineering
- A metamodel that defines the foundation classes of the method components
- A repository of reusable method components derived from the foundation classes of the metamodel
- A metamethod for constructing system architecture engineering methods by selecting, tailoring, and integrating method components from the MFESA repository.

The Quality Assessment of System Architectures and their Requirements (QUASAR) is a method for assessing the quality of system architectures and architecturally-significant quality requirements. QUASAR is based on the concept of requirements- and architecture-level quality cases consisting of:

- Claims – developers' assertions that the (a) architecturally-significant quality requirements are sufficiently complete, correct, consistent, etc. and (b) architecture is sufficiently complete and meets the architecturally-significant requirements
- Arguments – clear, compelling, and relevant developer arguments that sufficiently justify the assessor's belief in the developers' *claims* (e.g., architectural decisions, inventions, engineering trade-offs, assumptions, and associated rationales)
- Evidence – adequate, credible, and official substantiation supporting the developers' *arguments* (e.g., architectural diagrams, models, and documents).

D. *Patterns and Repository for Architecture Diagnostics and Optimization*

Our pattern language for architecture assessments of service-oriented enterprise systems [17] provides a procedural method framework for architecture assessment processes and questionnaire design. We organize and represent our architecture assessment patterns according to the structures of the architecture maturity framework SOAMMI [13], [18]: *Architecture Domains*, *Architecture Areas*, *Problem Descriptions* - associated with *Specific Goals*, *Solution Elements* that are connected to *Specific Practices* and *Related Patterns*, which are subsequent connections of applicable patterns within the pattern language.

Linking elements to specific practices of the SOAMMI framework indicate solutions for architecture assessments and improvements of service-oriented enterprise systems. This assessment and improvement knowledge is both verification and design knowledge, which is a procedural knowledge based on standards, best practices, and assessment experience for architecture assessments of service-oriented enterprise systems. It is therefore both concrete and specific for setting the status of service-oriented enterprise architectures, and helps to establish an improvement path for change.

We have identified and distinguished a set of 43 patterns as parts of a newly designed pattern language in the context of 7 Architecture Domains and 22 Architecture Areas. Even though our architecture quality patterns accord to the Specific as well as the Generic Goals and Practices of the SOAMMI framework, they extend these structures by navigable patterns [18], as part of an architecture assessment language. This pattern structure enables architecture quality assessors to navigate bi-directionally, to support both diagnostics and optimization processes, as well as to provide a clear link to questionnaires.

E. *Enterprise Architecture Governanace and Management*

Architecture Governance defines and maintains the Architecture Governance cycle [13]. It sets the abstract governance frame for concrete architecture activities within the enterprise or a product line development and specifies the following management activities: plan, define, enable, measure, and control. The second aim of Architecture Governance is to set rules for architecture compliance to internal and external standards. Enterprise and software architects are acting on a sophisticated connection path emanating from business and IT strategy to the architecture landscape realization for interrelated business domains, applications and technologies. Architecture Governance has to set rules for the empowerment of people, defining the structures and procedures of an Architecture Governance Board, and setting rules for communication. We specify architecture governance models for concepts such as: service strategy and life cycle management of software and system architecture artifact's state, service security, service testing and monitoring, service contracts, registries, service reuse, service ownership, definition and versioning.

VI. Conclusion

SmartLife applications are emerging as intelligent user-centered systems that will shape future trends in technology and communication. The development of such applications integrates web services, cloud computing, and big data management, among other frameworks and methods. The basic approaches within each field are already well known and used. However, such methods are not directly applicable and properly integrated for SmartLife applications. Existing approaches can be extended to exploit synergistic effects resulting from the SmartLife context. Technological evolution is also expected forming a feedback cycle from SmartLife scenarios to new technologies.

We have set up a transatlantic, multi-institutional research cooperation starting with this project, which would be extended to related areas as well as to student and academic exchanges and common publication efforts in conferences and journals. This paper described the framework and methodology for the research in progress. We explore SmartLife applications in the context of semantic-supported systems architectures and big data in cloud settings. Using a SmartLife application scenario, we investigate graph data management, fast big data, and semantic support through ontological modeling.

We have developed a prototype SmartLife application, WebAutoParts, to use as a test bed for our research project. We are exploring how the semantic and social data in the SmartLife scenario can be efficiently managed as big data, and how real-time analysis and updates can be efficiently performed on big graph data in cloud settings.

In addition, we have defined the ontological and architectural reference frameworks for our target SmartLife application, and are currently working on developing the ontological model for this application. Future work includes analyzing how the ontological and architecture models developed can be used to support semantic analysis and program comprehension of SmartLife applications. The models can be compared to and combined with other semantic modeling approaches to support development and maintenance of SmartLife applications.

Acknowledgments

Work described in this paper was partially supported by the University of West Florida Foundation under the Nystul Eminent Scholar Endowment, and the SOA Innovation Lab Germany.

References

[1] T. Reichherzer, E. El-Sheikh, N. Wilde, L. White, J. Coffey, and S. Simmons, "Towards intelligent search support for web services evolution: identifying the right abstractions", Proceedings of 2011 13th IEEE International Symposium on Web Systems Evolution (WSE), 30 Sept. 2011, pp. 53-58.

[2] J. Gray, A. Szalay, "Science In An Exponential World". Nature, , 23 March 2006, V. 440.23.

[3] Cheng, J., Ke, Y., and Ng, W.: Efficient query processing on graph databases. ACM Trans. Database Syst. 34, 1, Article 2, April 2009.

[4] Frischbier, S., Petrov, I.: Aspects of Data-Intensive Cloud Computing. From Active Data Management to Event-Based Systems and More, 2010, pp. 57-77.

[5] R. Sears, R. Ramakrishnan, "bLSM: a general purpose log structured merge tree", In Proc. of SIGMOD 2012.

[6] Open Group, "Service-Oriented Architecture Ontology", Technical Standard, The Open Group, 2010.

[7] J. Novak, and D. Gowin, "Learning How to Learn", Cambridge. University Press, New York, NY, 1984.

[8] J. W. Coffey and T. Eskridge, "Case Studies of Knowledge Modeling for Knowledge Preservation and Sharing in the U.S. Nuclear Power Industry", Journal of Information and Knowledge Management. 7(3), 2008, pp. 173-18.

[9] S. Bagui and R. Earp, (2012). "Database Design Using ER Diagrams", 2nd edition, Taylor and Francis, 2012.

[10] S. Bagui, "Mapping XML Schema to Entity Relationship and Extended Entity Relationship Models", International Journal of Intelligent Information and Database Systems, 3(4), 2007, pp. 325-345.

[11] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, and R. Metz, OASIS "Reference Model for Service Oriented Architecture" 1.0, OASIS Standard, 12 October, 2006.

[12] Open Group "SOA Reference Architecture", The Open Group, 2011.

[13] A. Zimmermann, H. Buckow, H.-J. Groß, O.F. Nandico, G. Piller, and K. Prott, "Capability Diagnostics of Enterprise Service Architectures using a dedicated Software Architecture Reference Model", IEEE-SCC2011: Washington DC – July 5-10, 2011, pp. 592-599.

[14] M. Behrendt, B. Glaser, P. Kopp, R. Diekmann, G. Breiter, S. Pappe, H. Kreger, and A. Arsanjani, "Introduction and Architecture Overview – IBM Cloud Computing Reference Architecture 2.0", IBM, 2011.

[15] A. Zimmermann, and G. Zimmermann, "Enterprise Architecture Ontology for Services Computing", SERVICE COMPUTATION 2012: Nice – France – July 22-27, 2012, ISBN 978-1-61208-215-8, pp. 64-69.

[16] D. G. Firesmith with P. Capell, D. Falkenthal, C. B. Hammons, D. Latimer, and T. Merendino, "The Method Framework for Engineering System Architectures", CRC Presstaylor & Francis Group, 2009.

[17] T. Erl, "SOA Design Patterns", Prentice Hall. 2009.

[18] A. Zimmermann, F. Laux, and R. Reiners, "A Pattern Language for Architecture Assessments of Service-oriented Enterprise Systems", PATTERNS 2012: Nice – France – July 22-27, 2012, ISBN 978-1-61208-158-8, 2011, pp. 7-12.