On the Runtime Models for Complex, Distributed and Aware Systems

Milan Zdravković*, Miroslav Trajanović*

* Laboratory for Intelligent Production Systems (LIPS),
Faculty of Mechanical Engineering, University of Niš, Niš, Serbia

milan.zdravkovic@gmail.com, miroslav.trajanovic@masfak.ni.ac.rs

***Abstract* – Recent developments in the area of Internet of Things increase the pressure on the feasibility of current architectures of the Enterprise Information Systems (EIS), in terms of their complexity, flexibility and interoperability in a pervasive computing world. The fact that EISs are today hosted by the growing diversity of platforms and devices, urges as to consider new concepts that would take into account rapid deployment and setup in any circumstances. This paper presents the discussion of model-driven architectures and proposes the concept of EIS design that is ontology-driven, persistence-neutral, runtime-model-based. These concepts are to some extent demonstrated in the case of OntoApp tool for ontology scaffolding.**

I. Introduction

The emergence of the ubiquitous computing technologies, such as Wireless Sensor Networks (WSN), Cyber-physical Systems (CPS) [1], Internet-of-Things (IoT) [2] and Future Internet Enterprise Systems (FinES) [3] is posing the new challenges to the traditional body-of-knowledge and practice in the design and development of Enterprise Information Systems (EIS). The increasing diversity and multiplicity of platforms where EISs are operating (taking into account different devices, e.g. sensors, processing devices and actuators) and lack of common, unifying standards and theories bring the distributed, federated, adaptable architectures, with so-called self-* properties into the focus of EIS developers and users.

The huge number of identifiable devices, used also on a sharing basis, is expected to become a commodity in the future, also providing a technology tool for emergence of so-called “sensing enterprise” [4]. Such diversity will pose tremendous challenges related to the interoperability issues. The new technological landscape, provided by the Future Internet systems will thus establish interoperability problems as critical and possibly consider the interoperability as an inherent capability of the future information systems.

In the recently submitted position paper, IFAC TC5.3 Technical Committee for Enterprise Integration and Networking of the International Federation for Automatic Control addressed the several research challenges of the systems interoperability, in attempt to define the future research directions towards so-called Next Generation Enterprise Information System (NG EIS). The following properties have been identified as critical for NG EIS: omnipresence, model-driven architecture, openness, dynamic configurability, multiplicity of identities, awareness/inclusive sensing and computational flexibility. Based on these properties, a generic, abstract architecture has been proposed, as illustrated on Fig.1.

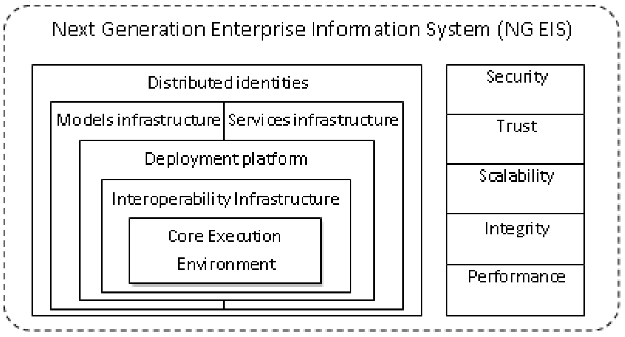


Fig.1 Abstract architecture of NG EIS, according to IFAC TC5.3

The main objective of the above proposal is to promote the research related to removing the complexity from the EIS design. In the ideal scenario, NG EIS will become a software shell, a core execution environment with the integrated interoperability infrastructure. Such an environment is foreseen as a highly flexible and scalable, deployable on any and every platform, using the external models and services infrastructure, exclusively or on a sharing basis. Finally, NG EIS will be highly extrovert system, represented by its multiple identities, e.g. as agents, UIs, services, etc.

In this position paper, we further build on that vision proposal by presenting the initial discussion on the possibilities to use conventional Model-Driven Engineering (MDE) tools and practices as enablers of NG EISs. When considering the objective of the technical unification of the core execution environments of NG EISs, we foresee that these environments will be driven by so-called runtime models. These runtime models will be the formal descriptions of the different aspects of the NG EIS design and operation, including data and information layer, business logics and UI. Finally, the objective of the work behind this paper is not to propose new formal models which will act as runtime models, but rather to rely on the vast number of existing domain and application ontologies.

II. Theoretical Background

The complete software development cycle can be viewed as a form of implicit modeling. For example, the problem analysis and identification of the solution is based on the experience of software architects, where this experience can be in fact considered as a model of knowledge. Then, this model of knowledge is specialized to the mental schemas of the architects.

Similar can be said also for the coding process. A code is also an implicit model, because it is built based on the specific language abstractions (syntax), software patterns and mental schemas of the coders who combine the syntax with the software patterns and experience to produce the code [5]. In this process, a gap between the problem and implementation definitions appears when "a developer implements software solutions to the problems by using abstractions that are at a lower level than those used to express the problem." [5]

The development process issues that contribute to the gap mentioned above are commonly addressed by Model-Driven Engineering (MDE) practices. MDE is a software development approach in which abstract models of software systems are systematically transformed to their specific implementations. MDE is driven and motivated by the growing complexity of software, which additionally increases the gap between the problem-level software abstractions (e.g. requirements) and its implementation.

Today, MDE approaches and practices are commonly addressed by using a Model Driven Architecture (MDA) [6]. MDA is a framework of MDE standards, launched and maintained by Object Management Group (OMG). It distinguishes between computation independent (CIM), platform independent (PIM) and platform specific (PSM) models. Main pillars of MDA are Meta Object Facility (MOF) language for defining the abstract syntax of modeling languages [7], UML [8] and Query, View, Transformation standard (QVT) for specifying PIM to PSM transformations [9].

Existing MDE tools and practices assume fixed viewpoints to one system. Now, each of these viewpoints needs also to consider the multiple identities of one system. The separation of concerns (e.g. functional, security, privacy, performance, etc.) in MDE approach needs to consider these identities.

Model transformations are considered as some of the key enablers for system interoperability. Despite the extensive results in the area of meta-modeling (e.g. MOF), the foundation for specifying transformations between models has not been built yet [10].

A. Formal Specification Techniques (FST)

One of the main problems of the current models is a lack of validation tools. Typically, the models of complex information systems are extensively large and in general, there exist no tools for querying and navigating them. More important, there exist no tools for their analysis. Thus, it becomes very difficult to maintain their consistency.

Formal Specification Techniques (FST) aim at restricting the modeling viewpoints, with objective to provide analysis, transformation and generation tools. A common approach is to translate a modeling view (e.g. a UML class model) to a form that can be analyzed using a particular formal technique [11]. This analysis can involve the consistency checking (for example, the relationships between the occurrences of the same software artifact in different viewpoints), completeness and dependability. Thus, reasoning on the formal specification of one system can be further used to prove that all actions will result in a discrete set of states, that some system properties are bounded, that error states are unreachable, etc.

Use of FST dates from the late seventies. Abrial et al [12] have proposed Z notation – a formal specification language for describing and modeling computing systems, based on Zermelo-Fraenkel set theory. Alloy has been developed [13] as a language for describing structural properties and their automatic semantic analysis. It is associated with an analyzer tool [14].

FST aims at facilitating so-called transformational programming or program transformation. The latter refers to an operation which transforms one computer program to another, which is "semantically equivalent to the original, relative to a particular formal semantics" [15]. The transformations are carried out incrementally, in manageable, controlled transformation steps which guarantee that the final software product will meet the initial specification. Although first concepts of program transformation has been defined in early seventies [16], the first exhaustive methodology was defined in scope of the Munich project CIP (computer-aided intuition-guided programming). That research included the "design of a wide-spectrum language specifically tailored to the needs of transformational programming, the construction of a transformation system to support the methodology, and the study of transformation rules and other methodological issues." [17].

Although current FST techniques are considered as self-sufficient, many authors addressed the problem of transforming widely accepted UML models to formal specification languages. Precise semantic characterizations of Object-Oriented modeling concepts have been introduced in 1997 [18]. The different tools have been developed to facilitate transformations of UML annotated class diagrams to complete Z [19] or Alloy [20] specifications. Csertdn et al [21] developed a transformation-based verification and validation environment for improving the quality of systems designed using the UML by automatically checking consistency, completeness, and dependability requirements.

1) Formal specification of business logic

Besides formal verification and validation, FST could also have significant role in semantically annotating the IS architecture, as well as the code. This can be achieved by correlating the specific FST formalisms to the lower level semantics of the domain ontologies, as well as the higher level semantics which is using generic, IS concepts, but is not bound to the specific domain.

For example, high level semantics is sometimes used to describe the businesses, e.g. by taking process perspective (BPEL). One of the examples of the lower level semantics are Business Process Patterns – formal and explicit descriptions of the generalized designs – best practices for business in a given application domain [22]. A Domain Specific Model (DSM) for business logic of information systems is proposed, based on the analysis of the modeling concepts of visual behavioral modeling languages [23]. DSM is considered as abstract business logic model (called Amabulo meta-model), combining process, state and structural perspectives.

B. Runtime models

Typically, runtime models are considered as assets which are used to monitor and verify particular aspects of the runtime behavior of the information system [24][25]. It is foreseen that the runtime models will be used by the agents responsible for managing the runtime environment, and for adapting and evolving the software during runtime. Hence, the models are considered as interfaces between running systems and change agents, where a change agent can be a human developer or a software agent [5].

Research on the runtime models is still in its infancy; however, the foreseen opportunities are beyond significant. France and Rumpe [5] predicted the possibilities of system users to observe and understand system behavior, adaptation agents to detect the need for adaptations and perform these, change agents to handle errors and introduce new features, all by using runtime models. It is obvious that these assumptions are inherited from the theory of adaptive, self-managed systems which specifically deal with change processes [26].

In the above described context, runtime models are expected to evolve the current practices of MDE, from design, implementation and verification stages of development of the EIS, to their actual execution. The evolution of models and associated approaches to a change management is considered only as a first step, towards the vision of EIS's as shells which are actually executing models, which embed all application aspects, including data and information, business logics and user interface.

With the emergence of semantic technologies, some initial works on developing ontology-driven systems, using formal models, expressed in RDF/OWL, have been carried out.

C. Ontology-driven systems

Today, ontologies are increasingly used to facilitate a process of software design. However, in great most of the cases, their use is related to maintaining different schemas (with increased expressiveness when comparing to conventional semi-structured data approach, such as XML) and to providing the formal foundation to conventional MDE environments, thus enabling their verification.

Although the term of ontology-driven information system, as a system that make use of formally defined ontologies, was coined by Guarino in 1998 [27], the use of conceptual schemas to represent the knowledge about specific application domain was proposed as early as in the seventies [28].

The common use of ontology for information systems is related to facilitating cross-domain explanation and understanding of invariants of one domain. There are opinions that it should not be confused with the different conceptual schemas (e.g. ER, UML, OMT) used in systems' modeling, as latter involve specification of a meaning of these invariants in the different dimensions [29].

Here, we would like to highlight the difference between ontology-based and ontology-driven software. In the former case, even though that sometimes ontologies play the central role, a large amount of business logic is still implicitly contained in a source code. The latter case fully corresponds to runtime models paradigm, where all data structures and business logic are formally described in ontology and then interpreted by the shell software at runtime. We use the notion of "shell software" because it can be then considered as a model execution platform.

Currently, there are only few works that use the ontologies as runtime application models. When considered as data schemas, ontologies can be used to drive software which would enable managing its individuals. Such software is then no more than a tool for ontology browsing and concept instantiating. Ontology Based Information System (OBIS) [30] is an example of such an approach.

OntoWebber System Architecture [31] enabled web portal management, where ontological framework is used to integrate and semantically align different data sources in order to generate a web portal. To a minor extent, it also addressed business logic, by enabling formal description and correspondent on-demand operation of few business rules, related to personalization and web site maintenance.

One of the proposed approaches to ontology-driven software [32] considers the interaction flow as a key artifact of runtime ontology. This flow consists of the actions of registration of the specific event occurrence, its categorization, identification of the situation that matches the categorized event and consequent execution of one or more tasks (including interpretation of a business domain model to generate recommendation for some of these tasks). In fact, such conceptualization can be used to formalize business rules and is thus useful for further development of ontology-driven software paradigm.

1) Ontology Scaffolding

Ontology scaffolding is an approach in which a basic application, with so-called CRUD (Create, Read, Update and Delete) functionality is generated in design or run time, based on a specified model.

Scaffolding approach became popular with the development of MVC (Model-View-Controller) frameworks and is typically related to using database schema to create scaffolds.

D. NoSQL databases

One of the critical objectives of the design of the future shell software that will execute runtime models is related to a full independence, relative to the agreements and sometimes, compromises on the use of the persistence layer. Currently, NoSQL databases are the best educated guess for facilitating such an approach, characterized by the two advantages over traditional relational database systems: flexibility and performance.

The flexibility of NoSQL is arising from the fact that it does not have to adhere to schema definitions, which typically correspond to data model of application which is using those. Instead, information is stored in semi-structured way, by using key-value pairs, documents, graphs or wide-column stores. The light structure of the storage formalisms contributes to the high horizontal scalability of NoSQL databases. Namely, unstructured data can be more easily stored across multiple processing nodes, because it does not follow complex data structures and it avoids join operations.

When comparing the SQL and NoSQL databases, the latter are highly preferred options for large data sets and hierarchical data structures. SQL databases still outperform NoSQL in facilitating complex transactional applications. Finally, native support to ACID makes relational databases superior to NoSQL when reliability and consistency are considered.

III. OntoApp Tool

OntoApp is ontology scaffolding tool which generates CRUD functionality in runtime, based on the specified ontology – RDF/XML file. It is a PHP web application, using RDF API for PHP [33] for ontology interpretation and Neo4JPHP API for storage. It uses Neo4J graph database [34] for storing instances. Thus, it is persistence neutral in the sense that it does not adhere to a specific structure of the database as an implementation condition.

In this paper, we refer to the case of ontology-driven project management application. The application is implemented by using OntoApp with the simple project management ontology. The UML representation of the portion of project.owl ontology is illustrated on Figure 2.

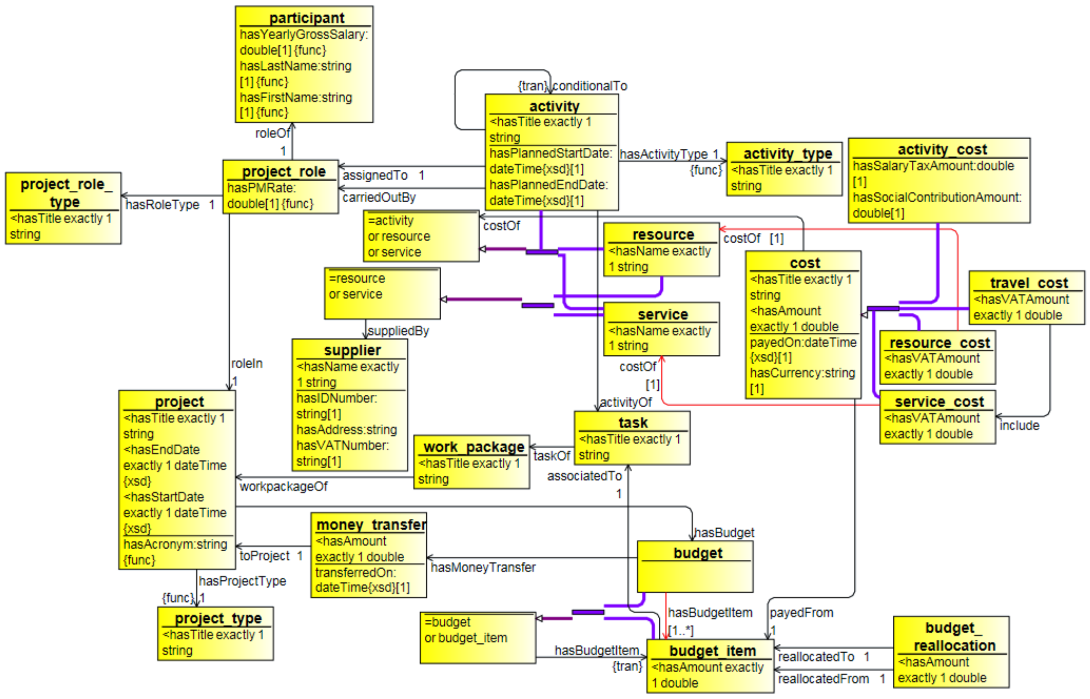

participant
hasYearlyGrossSalary: double[1] {func}
hasLastName:string [1] {func}
hasFirstName:string [1] {func}
activity
<hasTitle exactly 1 string
hasPlannedStartDate: dateTime{xsd}[1]
hasPlannedEndDate: dateTime{xsd}[1]
{tran} conditionalTo
project_role_type
<hasTitle exactly 1 string
project_role
hasPMRate: double[1] {func}
hasRoleType 1
roleOf 1
assignedTo 1
carriedOutBy
hasActivityType 1 {func}
activity_type
<hasTitle exactly 1 string
activity_cost
hasSalaryTaxAmount:double [1]
hasSocialContributionAmount: double[1]
=activity or resource or service
costOf
resource
<hasName exactly 1 string
costOf [1]
cost
<hasTitle exactly 1 string
<hasAmount exactly 1 double
payedOn:dateTime {xsd}[1]
hasCurrency:string [1]
=resource or service
suppliedBy
service
<hasName exactly 1 string
travel_cost
<hasVATAmount exactly 1 double
resource_cost
<hasVATAmount exactly 1 double
service_cost
<hasVATAmount exactly 1 double
include
supplier
<hasName exactly 1 string
hasIDNumber: string[1]
hasAddress:string
hasVATNumber: string[1]
roleIn 1
project
<hasTitle exactly 1 string
<hasEndDate exactly 1 dateTime {xsd}
<hasStartDate exactly 1 dateTime {xsd}
hasAcronym:string {func}
costOf [1]
activityOf
task
<hasTitle exactly 1 string
work_package
<hasTitle exactly 1 string
taskOf
associatedTo 1
workpackageOf
hasBudget
money_transfer
<hasAmount exactly 1 double
transferredOn: dateTime{xsd}[1]
toProject 1
hasMoneyTransfer
budget
hasProjectType {func} 1
project_type
<hasTitle exactly 1 string
=budget or budget_item
hasBudgetItem [1..*]
payedFrom 1
hasBudgetItem {tran}
budget_item
<hasAmount exactly 1 double
reallocatedTo 1
reallocatedFrom 1
budget_reallocation
<hasAmount exactly 1 double

Fig.2 UML Representation of the project.owl ontology

The different views to the different concepts of example ontology in OntoApp are illustrated on Figure 3.

OntoApp interprets the formal model, expressed as RDF/XML ontology to generate CRUD functionality on the set of concepts, as specified in ontology. Based on the formal definition of each of the concepts, the form is being generated in a runtime and used to define an instance of the given concept, which is then stored in Neo4J database as a graph node, with a label corresponding to the name of the concept.

The data properties are defined as properties of a node, while each of the instantiated object property correspond to the relationship being established in a graph database between the specific node and another existing node in a domain of the object property.

Furthermore, the generated form implements certain validation rules which are determined based on the formal definition of the concept which instances are being created. OntoApp interprets the formal restrictions expressed as anonymous parent concepts – necessary conditions for a given concept. These formal restrictions are defined as value (owl:allValuesFrom, owl:someValuesFrom) and cardinality (owl:cardinality, owl:minCardinality, owl:maxCardinality) constraints of both data and object properties.

A. Interoperability as an inherent property of OntoApp

One of the key benefits of the formal runtime-model-driven applications is that they are inherently interoperable.

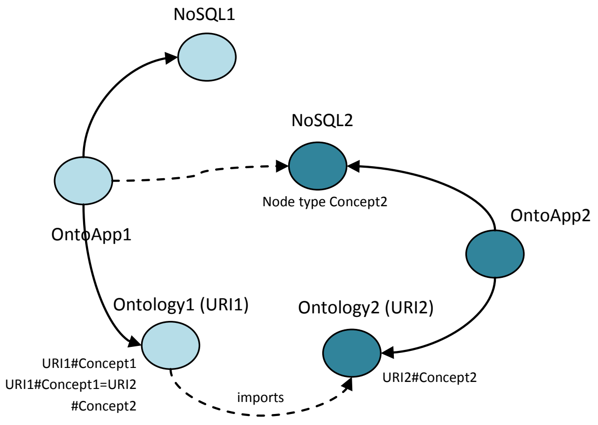


Fig.3 Illustration of the interoperability as inherent property of OntoApp

This inherent interoperability arises from the core of the approach. Namely, each OntoApp system comprises of three assets, natively distributed (see Fig.3): OntoApp, ontology and NoSQL database. OntoApp execution environments (OntoApp1, OntoApp2) are installed on the different platforms. Each of the environments is driven by one of the respective ontologies (Ontology1, Ontology2) with specified Uniform Resource Identifiers (URI1, URI2). Then, the execution environments are using ontologies to create and manage graphs, stored in respective NoSQL databases (NoSQL1, NoSQL2).

When Ontology1, driving OntoApp1 environment imports Ontology2, the concepts of the latter can be instantiated (nodes created and managed) in NoSQL2, by using OntoApp1. This is possible only for concepts of Ontology2, which are annotated with the connection strings, corresponding to the location and authentication of NoSQL2 database; and defined access rights.

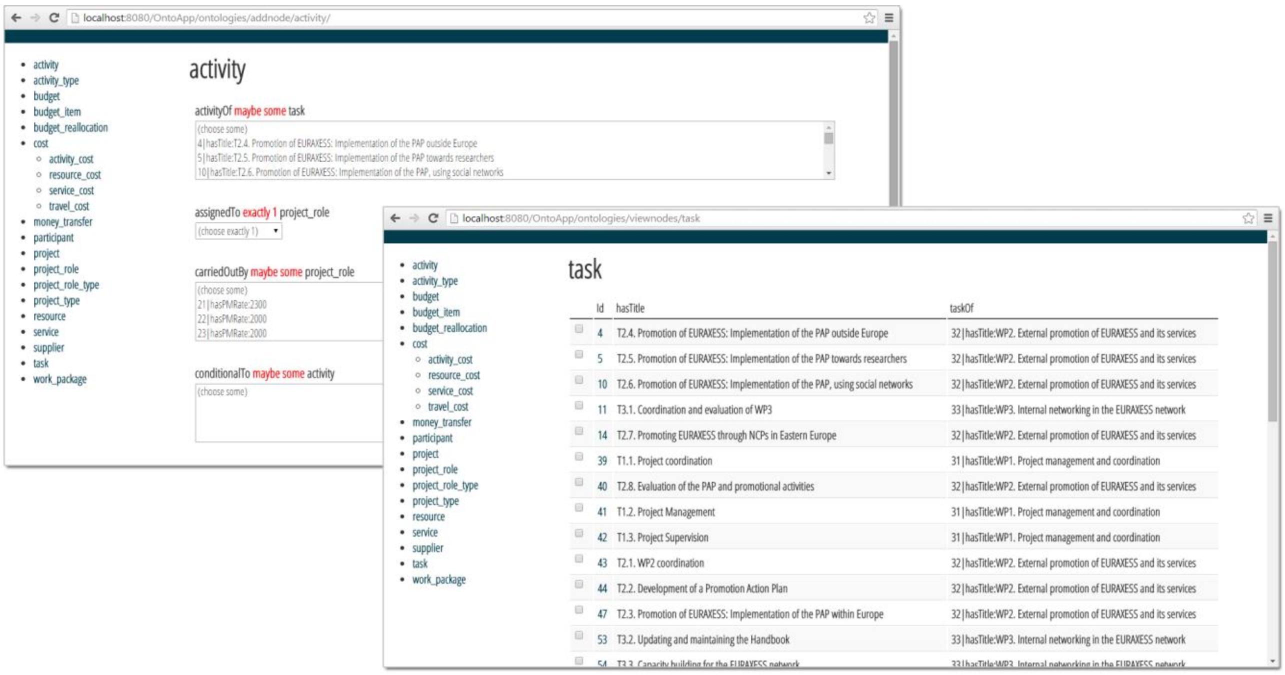
When logical equivalence relationship is established between the different concepts in source (Ontology1) and imported ontology, then OntoApp1 can be enabled with a centralized access to the distributed repository of information objects, thus enabling for example, integrated reporting, bulk processing, etc.

Finally, using other logical relationships to connect the different concepts from the different ontologies enables the construction and maintenance of the federated objects, whose different attributes are stored across multiple NoSQL repositories.

B. *Future Development*

Currently, OntoApp tool development aims at investigating possibilities to: 1) further customization of CRUD functionality, in terms of more closely adhering to the actual specific users' needs; and 2) modeling and embedding elements of the business logic into application. In order to make this possible, the helper ontology is being developed, that will enable formal definition of the rules related to restricting the accessibility to the specific concepts, instances or sub-graphs and business rules.

Access restriction is being implemented in two ways. First, simple user administration model is being embedded, based on its formal definition in the helper ontology. It defines user instances (stored in helper ontology) and assignments of CRUD rights, on the specified concepts of the imported ontology – runtime model. Second, context restriction is being implemented. It enables restriction of CRUD rights based on the selected instances – enabling access to all instances (nodes) from or to which a graph can be traversed.



localhost:8080/OntoApp/ontologies/addnode/activity/

- activity
- activity_type
- budget
- budget_item
- budget_reallocation
- cost
  - activity_cost
  - resource_cost
  - service_cost
  - travel_cost
- money_transfer
- participant
- project
- project_role
- project_role_type
- project_type
- resource
- service
- supplier
- task
- work_package

activity

activityOf maybe some task
(choose some)
4|hasTitle:T2.4. Promotion of EURAXESS: Implementation of the PAP outside Europe
5|hasTitle:T2.5. Promotion of EURAXESS: Implementation of the PAP towards researchers
10|hasTitle:T2.6. Promotion of EURAXESS: Implementation of the PAP, using social networks

assignedTo exactly 1 project_role
(choose exactly 1)

carriedOutBy maybe some project_role
(choose some)
21|hasPMRate:2300
22|hasPMRate:2000
23|hasPMRate:2000

conditionalTo maybe some activity
(choose some)

localhost:8080/OntoApp/ontologies/viewnodes/task

- activity
- activity_type
- budget
- budget_item
- budget_reallocation
- cost
  - activity_cost
  - resource_cost
  - service_cost
  - travel_cost
- money_transfer
- participant
- project
- project_role
- project_role_type
- project_type
- resource
- service
- supplier
- task
- work_package

task

Id	hasTitle	taskOf
4	T2.4. Promotion of EURAXESS: Implementation of the PAP outside Europe	32|hasTitle:WP2. External promotion of EURAXESS and its services
5	T2.5. Promotion of EURAXESS: Implementation of the PAP towards researchers	32|hasTitle:WP2. External promotion of EURAXESS and its services
10	T2.6. Promotion of EURAXESS: Implementation of the PAP, using social networks	32|hasTitle:WP2. External promotion of EURAXESS and its services
11	T3.1. Coordination and evaluation of WP3	33|hasTitle:WP3. Internal networking in the EURAXESS network
14	T2.7. Promoting EURAXESS through NCPs in Eastern Europe	32|hasTitle:WP2. External promotion of EURAXESS and its services
39	T1.1. Project coordination	31|hasTitle:WP1. Project management and coordination
40	T2.8. Evaluation of the PAP and promotional activities	32|hasTitle:WP2. External promotion of EURAXESS and its services
41	T1.2. Project Management	31|hasTitle:WP1. Project management and coordination
42	T1.3. Project Supervision	31|hasTitle:WP1. Project management and coordination
43	T2.1. WP2 coordination	32|hasTitle:WP2. External promotion of EURAXESS and its services
44	T2.2. Development of a Promotion Action Plan	32|hasTitle:WP2. External promotion of EURAXESS and its services
47	T2.3. Promotion of EURAXESS: Implementation of the PAP within Europe	32|hasTitle:WP2. External promotion of EURAXESS and its services
53	T3.2. Updating and maintaining the Handbook	33|hasTitle:WP3. Internal networking in the EURAXESS network
54	T3.3. Capacity building for the EURAXESS network	33|hasTitle:WP3. Internal networking in the EURAXESS network

Fig.4 OntoApp

Another aspect of customization is related to enabling users to customize the layout of the app by themselves, e.g. to rearrange order of properties in add/edit forms, build the custom menus, etc.

The work on modeling and embedding elements of the business logic into app is still in initial phase and thus, it will not be considered in this paper.

IV. Conclusion

Current research on runtime models consider their application on the traditional EIS architecture, where the runtime models are used by the change agents. Although this work is still in initial phase, the research on the runtime models with full functional coverage and respective execution environments is expected to gain the attention of the research community, due to widening gap between the traditional approaches to EIS design and development and rapidly growing diversity of identifiable processing platforms.

Besides the significant potential impact, related to dramatically improved flexibility and facilitation of so-called self-* properties of EIS, e.g. self-management, self-optimization, etc., the use of runtime models and their execution environments is also expected to simplify and accelerate innovation, since they facilitate rapid verification and validation of new concepts and ideas.

In its current shape, OntoApp is a simple and yet functional tool for managing information in a specific domain, where this domain is formally described by using the ontology. OntoApp is inherently interoperable, since it relies exclusively on the semantic model, namely the ontology, which concepts can be made logically correspondent to the concepts of the other domain ontologies.

It is expected that further development in customization only would contribute to its usefulness beyond the limits of the academic exercise. Further research and implementation of the other different aspects, such as business rules and reporting is expected to bring the clear evidence on the concept of model execution environments, as a key feature of the future NG EIS.

ACKNOWLEDGMENT

The work presented in this paper was supported by the Ministry of Education and Science of the Republic of Serbia (project No. III41017).

REFERENCES

[1] Lee, E., Cyber Physical Systems: Design Challenges. Technical Report No. UCB/EECS-2008-8, 2008, University of California, Berkeley.

[2] Ashton, K., That 'Internet of Things' Thing, in the real world things matter more than ideas. RFID, http://www.rfidjournal.com/articles/view?4986, 2009.

[3] FInES Future INternet Enterprise Systems - Research Roadmap 2025. 2012.

[4] Santucci, G., C. Martinez, and D. Vlad-Câlcic The Sensing Enterprise. 2012.

[5] France, R., & Rumpe, B. (2007). Model-driven Development of Complex Software: A Research Roadmap. Proceedings of Future of Software Engineering (FOSE 07) (pp. 37 - 54). Washington, DC, USA: IEEE Computer Society.

[6] Soley, R. M., Frankel, D., Mukerji, J., & Castain, E. (2001). Model Driven Architecture - The Architecture of Choice. OMG.

[7] OMG Adopted Specification ptc/03-10-04. The Meta Object. OMG.

[8] (2005). The Object Management Group - UML 2.0: Superstructure Specification. Version 2.0.

[9] QVT-Merge Group 1.8. Revised submission for MOF 2.0. OMG.

[10] Czarnecki, K., & Helsen, S. (2006). Feature-based survey of model transformation approaches. IBM Systems Journal , 45 (3), 621 - 645.

[11] McUmber, W. E., & Cheng, B. H. (2001). A general framework for formalizing UML with formal languages. Proceedings of the 23rd International Conference on Software Engineering (pp. 433-442). Washington, DC, USA: IEEE Computer Society.

[12] Abrial, J.-R., Schuman, S. A., & Meyer, B. (1980). A Specification Language. In A. M. Macnaghten, & R. M. McKeag, On the Construction of Programs. Cambridge University Press.

[13] Jackson, D. (2002). Alloy: a lightweight object modelling notation. ACM Transactions on Software Engineering and Methodology , 11 (2), 256 - 290.

[14] Jackson, D., Schechter, I., & Shlyakhter, I. (2000). Alcoa: the Alloy constraint analyzer. Proceedings of the 2000 International Conference on Software Engineering (pp. 730 - 733). Limerick: IEEE.

[15] Martin, W. 1989 Proving Program Refinements and Transformations. PhD Thesis, Oxford University

[16] Cheatham, T. E., and Wegbreit, Ben. A Laboratory for the Study of Automating Programming Proc AFIPS 1972 Spring Joint Computer Conf., 1972

[17] Bauer, F. L., Moller, B., Partsch, H., & Pepper, P. (1989). Formal program construction by transformations-computer-aided, intuition-guided programming. IEEE Transactions on Software Engineering , 15 (2), 165 - 180.

[18] Schroff, M., & France, R. B. (1997). Towards a formalization of UML class structures in Z. Proceedings of the Twenty-First Annual International Computer Software and Applications Conference (pp. 646 - 651). Washington, DC: IEEE.

[19] Dupuy, S., Ledru, Y., & Chabre-Peccoud, M. (2000). An Overview of RoZ : A Tool for Integrating UML and Z Specifications. Advanced Information Systems Engineering. Lecture Notes in Computer Science. 1789, pp. 417 - 430. Springer Berlin Heidelberg.

[20] Anastasakis, K., Bordbar, B., Georg, G., & Ray, I. (2010). On challenges of model transformation from UML to Alloy. Software & Systems Modeling , 9 (1), 69 - 86.

[21] Csertdn, G., Huszerl, G., Majzik, I., & Pap, Z. (2002). VIATRA - visual automated transformations for formal verification and validation of UML models. Proceedings of 17th IEEE International Conference on Automated Software Engineering (pp. 267 - 270). IEEE.

[22] Barros, O. (n.d.). A Novel Approach to Joint Business and Information System Design.

[23] Brückmann, T., & Gruhn, V. (n.d.). A Domain Specific Model to Support Model Driven Development of Business Logic.

[24] Bencomo, N., Blair, G., & France, R. (2007). Summary of the Workshop Models@run.time at MoDELS 2006. Models in Software Engineering, Lecture Notes in Computer Science. 4364, pp. 227 - 231. : Springer.

[25] Bencomo, N., France, R.B., Cheng, B.H.C., Asmann, U. (Eds) Models@runtime: Foundations, Applications and Roadmaps. Lecture Notes in Computer Science, Vol.8378, Springer

[26] Bradbury, J. S., Cordy, J. R., Dingel, J., & Wermelinger, M. (2004). A Survey of Self-Management in Dynamic Software Architecture Specifications. Proceedings of the International Workshop on Self-Managed Systems (pp. 28 - 33). Newport beach, CA, USA: ACM.

[27] Guarino, N. (1998). Formal Ontology and Information Systems. In N. Guarino, Formal Ontology in Information Systems (pp. 3 - 15). Amsterdam, Netherlands: IOS Press.

[28] Tsichritzis, D., & Klug, A. C. (1978). The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Dabatase Management Systems. Information Systems , 3 (3), 173 - 191.

[29] Fonseca, F., & Martin, J. (2007). Learning The Differences Between Ontologies and Conceptual Schemas Through Ontology-Driven Information Systems. Journal of the Association for Information Systems , 8 (2), 129 - 142.

[30] Zviedris, M., Romane, A., Barzdins, G., & Cerans, K. (2014). Ontology-Based Information System. Semantic Technology. Lecture Notes in Computer Science (pp. 33 - 47). Seoul, South Korea: Springer International Publishing.

[31] Jin, Y., Decker, S., & Wiederhold, G. (2001). OntoWebber: Model-Driven Ontology-Based Web Site Management. Proceedings of SWWS'01, The first Semantic Web Working Symposium, (pp. 529 - 547). Stanford, CA, USA.

[32] Hammitt, L. C., & Beckert, J. (2007). Patent No. US7200563 B1. USA.

[33] Oldakowski, R., Bizer, C. 2004. RAP: RDF API for PHP

[34] Miller, J.J. 2013. Graph Database Applications and Concepts with Neo4j. In Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA March 23rd-24th, 2013