An Ontological Model for Generating Complete, Form-based, Business Web Applications

Daniel Strmečki[1], Ivan Magdalenić[2]
Faculty or Organization and Informatics, University of Zagreb, Varaždin, Croatia

***Abstract*—This paper presents an ontological model for specifying and automatically generating complete business Web applications. First, a modular and expandable ontological model for specifying form-based, business Web applications is developed and presented. Next, the technology used for transforming the ontological specification to Java executable code is explained. Finally, the results of applying the proposed model for specifying and generating an order management application are presented. Results showed that the application of an ontological model in a generative programming approach increases the level of abstraction. This approach is especially suitable for development of software families, where similar features are reused in multiple products/applications.**

Keywords—Ontology; model; generative; automatic; programming; development

I. INTRODUCTION

Software reuse is an important area of software engineering discipline, as it can increase software productivity and quality. However, factors like deadlines, budget, technology, architecture and the level of knowledge and experience also need to be taken into account. Raising the abstraction level is the most commonly used approach to increase software reuse. By encapsulating knowledge about lower level operations, developers and engineers can think in terms of higher level concepts, thus saving effort and time [1], [2].

Automatic programming is a software engineering discipline that automates the lower level process. Its main goal is to enable developers to operate on higher abstraction levels by making machines do parts of the programming work. Even after more than 60 years since its appearance, there are still no universal solutions for software development automation. However, it must be acknowledged that the discipline has dramatically influenced on improvement of software developers' productivity. A high level of automation in software development can nowadays be achieved in some domains, but the dream of enabling general-purpose, full development automation still remains unrealized [3].

Generative programming is a sub discipline of automatic programming that uses generators to facilitate the process of application development. A generator is piece of software that takes a higher-lever specification of another piece of software and produces its implementation [4]. A domain model is used to provide mapping between the problem space and solution space. Problem space in generative programming refers to a set of software features to be implemented, while solution space implies the implementation abstractions contained in the specification. A generator maps the two spaces by using a specification to yield the corresponding implementation [5].

Ontologies were initially applied in software development only to store data and their semantic meaning, but nowadays they are used in all phases of software development lifecycle. Even a new discipline Ontology-Driven Software Engineering arose, where ontologies are used to perform a majority of operations in software development [6]. Several authors found ontologies suitable and helpful in performing tasks like description of specification documents, formal representation of requirements, semantic description of services/components, domain model generation, test cases generation and executable code generation [7]–[16]. In this paper we present and apply an ontological model for generating business Web applications. We applied ontologies for both modeling and specifying a complete, form-based application. The ontological specification is then used as an input to code generators, which can produce fully functional Web applications.

This paper is organized as follows. Section 2 describes the related work. It describes and discusses models and frameworks used for the same purpose as the one presented here. Section 3 introduces a modular ontological model for modelling and generating business Web applications. Section 4 describes how the ontological model and its specification are used for generating complete, form-based, applications. Section 5 provides an example application of the developed ontological model. An order management business Web application is ontologically specified and generated. Section 6 presents our conclusions.

II. RELATED WORK

In their approach named Ontology Driven Architecture for Software Engineering authors Bossche et al. proposed the usage of ontologies for forming a knowledge continuum between business and IT. First, the business representatives work closely with domain modelling experts to build a formal business model. This enables the business to formalize their specification and forces them to make requirements explicit. Next, once the ontology is formed, a transformation from ontology to source code is done automatically. For this purpose, authors used a platform based on a strongly typed logic programming language Mercury and a set of tools and libraries called Hedwig [17]. They pointed out a number of advantages brought by the use of ontologies in automated programming, but it remains unclear what ontologies were used and how were they mapped to source code generators.

Authors Tang et al. developed a Web Information System auto-construction Environment, which is also a platform for automatic source code generation based on ontologies. Their platform uses a predefined ontology to specify user requirements and provides graphical tools for ontology construction and drawing user interfaces. It is based on three tools: builder, mapper and generator. The builder tool helps users construct the domain and behavior ontologies. The mapper tool provides methods to automatically transform behavior operations to backend database access code. The generator tool, as the name suggests, generates the source code from ontologies [18]. Since the platform generates program code automatically from predefined ontologies, it is not possible to add new ontological or user interface elements without extending the platform itself.

Semantic Web Builder is an agile development platform for Web application development, proposed by authors Solis et al. In their platform requirements are modeled using ontologies and the infrastructure is then automatically generated. The source code is however generated in two layers. A base layer is automatically generated and should never be modified. For implementation of specific functionality, an extended layer is also provided [19]. This presents the main disadvantage of their approach, as custom code needs to be developed manually for every specific functionality.

In this paper we will present an approach for generating complete Web applications from ontologies. We will explain how the ontological model fits into a source code generation platform, built in Java using only publicly available tools and frameworks. We will provide public access to the ontological model and the source code of the generators we have developed. This will make it possible for anyone to extend the model and generators, with their custom tweaks or new functionality. The goal of the proposed platform is to be able to model and generate any new or specific features trough ontologies, so there is no custom code and all features can be easily reused in future work. We rely on ontologies for modeling complete systems, generate fully functional applications and avoid repetitive operations, including any code or specification copy-pasting. The presented approach is limited to working with form-based Web application and relational databases.

III. Ontological Model

Given that we wanted to achieve a high level of reusability and expandability of the initially created ontological model, we decided to take a modular approach in the model creation. We defined five initial ontologies, where each is focused on one subdomain: 1) Requirements ontology, 2) Repository ontology, 3) Web forms ontology, 4) Web components ontology and 5) User interface ontology. Requirements ontology contains classes and attributes defining the project, client and requirements, including epics, user stories and acceptance criteria. Repository ontology contains classes and attributes defining a relational database, its tables and columns. In the initial version, only relational databases are supported, but other classes supporting different types of repositories could be developed in the future. Web forms ontology defines all forms available to the user through the application, as well as how forms are grouped together and displayed in the main menu. Web components ontology contains definitions of components and listeners. Components are further divided into input fields, action buttons, pagination tables, etc. The input fields are divided into the standard input field types we use on modern Web forms, such as text input fields, numeric input fields, select fields, combo fields, date input fields, etc. User interface ontology contains classes that define layout attributes and positioning of the components on the screen. *Fig. 1* shows the connections between the mentioned ontologies and how they fit into a complete ontological model for specifying form-based Web applications. The hierarchy of the complete ontological model is shown in *Fig. 2*. The complete model consists of 2300+ axioms, 30+ classes and 80+ types of properties, from which 37% are object properties and the rest are data properties.

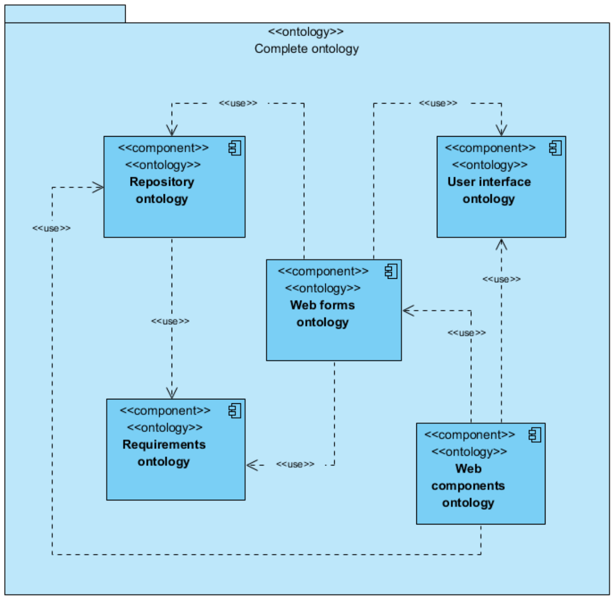
<<ontology>>
Complete ontology
<<use>>
<<use>>
<<component>> <<ontology>> Repository ontology
<<component>> <<ontology>> User interface ontology
<<use>>
<<component>> <<ontology>> Web forms ontology
<<use>>
<<use>>
<<use>>
<<component>> <<ontology>> Requirements ontology
<<use>>
<<component>> <<ontology>> Web components ontology

Fig. 1. Ontological Model Components.

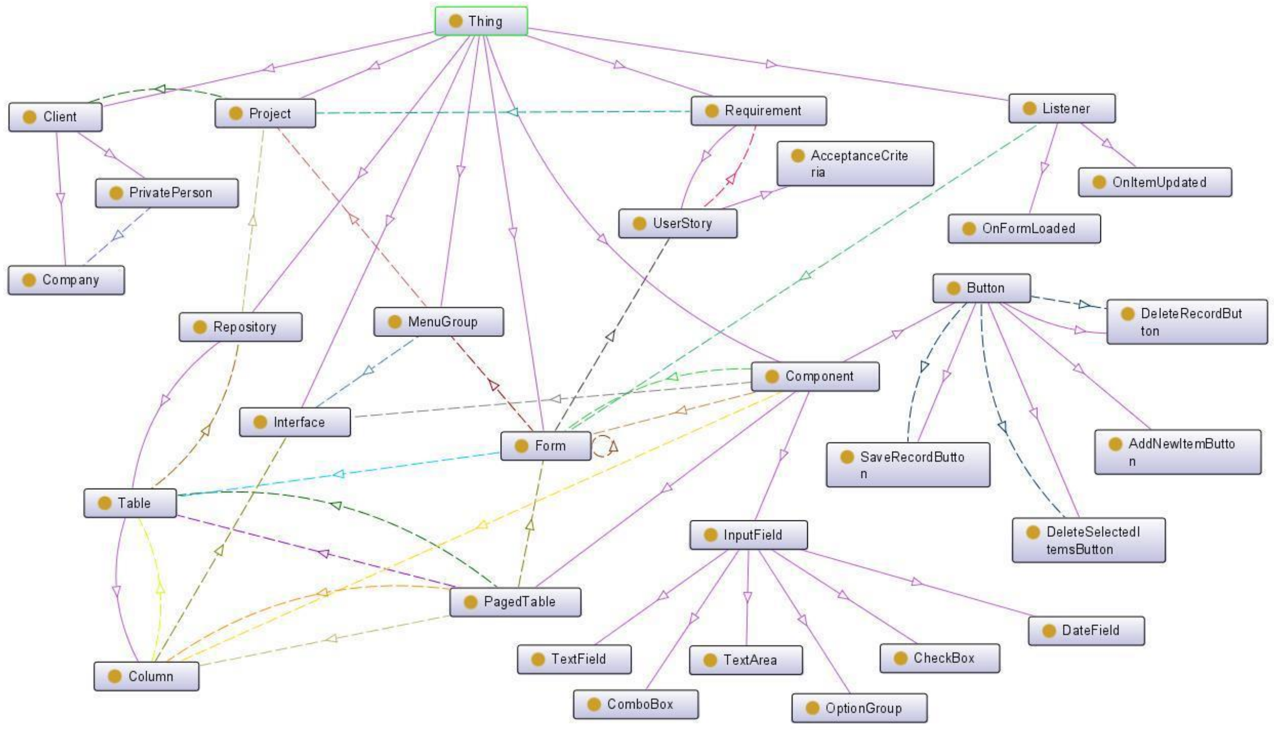

Fig. 2. Complete Ontological Model.

IV. Code Generation

The language of choice for development of code generators is Java, simply because it is a very popular object-oriented language, in which the author is working professionally for several years. Java also has a very good support for working with code templates, Web components and ontologies. Apache Jena is the framework we used for fetching the ontological model structure and querying the ontology in order to retrieve specification. We also used Apache Velocity framework for templating Java and HTML/CSS code. Velocity allows developers to use a simple template language with access to reference objects defined in the Java code. Next, we used Vaadin Web framework which allowed us to use a large number of free, out-of-the-box Web components. Vaadin uses a component-based approach for rapid development of user interfaces for Java Web applications. This is very convenient, because we want to focus on development of code generators, not user interface components. Finally, we used Hibernate framework as our object-relational mapping framework of choice. Hibernate enables us to define database entities as Java classes and is able to generate the database schema automatically.

The purpose of the ontological model and specification is to provide an input for code generation. As presented in *Fig. 3*, a code generator first fetches the ontological model vocabulary and then the specific application specification that needs to be generated. The ontological model vocabulary is represented in the generators source code by a single Java class for each ontology. These classes are automatically generated using a tool called Schemagen, provided with Apache Jena framework. We use these generated classes to retrieve data from the ontological model. Application specification data is retrieved from ontology by running SPARQL queries from generators Java code. Code generators map the specification of forms, components and database entities to appropriate Velocity code templates. This process results in generated files/classes of the final application. Once all the files are generated, the generation process results in a complete, fully functional Java Web application/project. The application code can be compiled and installed on an appropriate Java web server like Jetty or Tomcat. Besides the Web server, it is also necessary to provide a connection from the server to a relational database. We used MySQL database in our example application and Hibernate helped us to automatically create the database schema. Once the infrastructure is set up and the application is successfully installed, users can access the generated application forms through their Web browser.

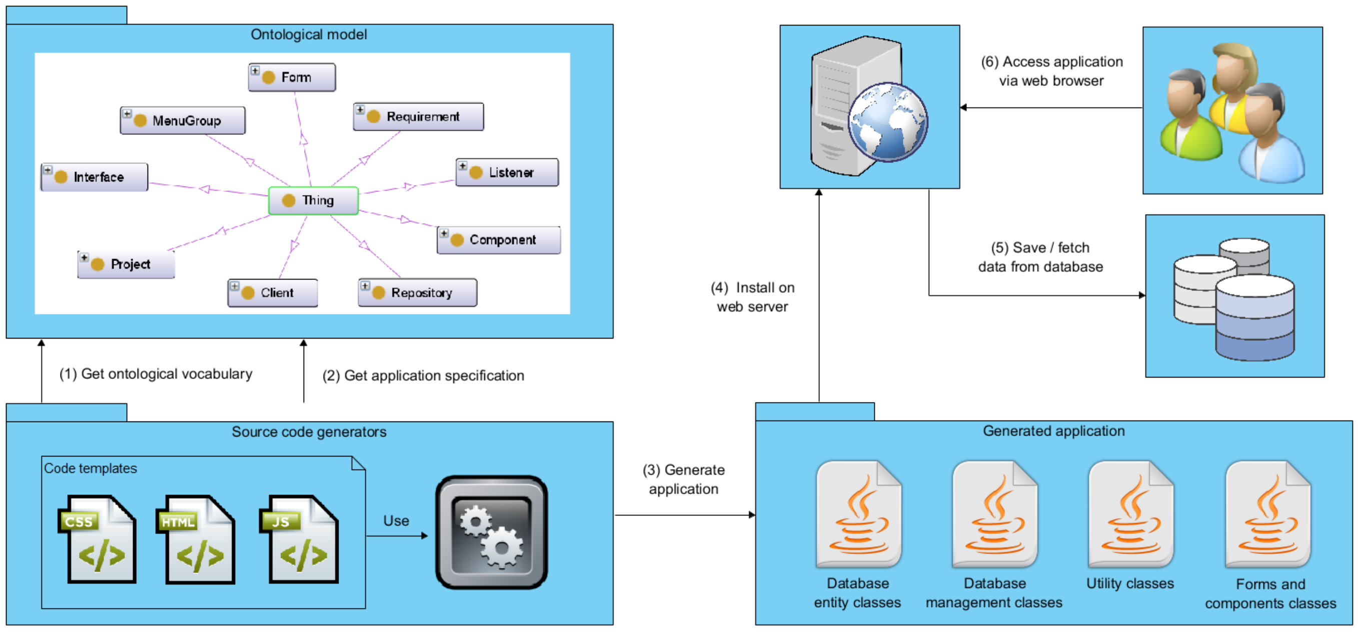

Fig. 3. Using an Ontological Model to Generate Executable Code.

V. Application

As an example application we ontologically specified and generated an order management application. This sample application contains a total of 18 Web forms including forms for managing addresses, articles, contacts, states, references, Incoterms, legal persons, private persons, orders and tax schemas. It is important to note that these are not only simple CRUD forms and that some of them contain complex relations. For example, it is possible to add items to existing orders and connect them to multiple referenced documents. When adding new items the total tax and payable amount get updated automatically. The generated forms contain almost 7.000 lines of Java code. The complete generated project resulted in 52 generated files, 10.000+ lines of code, 5500+ statements, 600+ methods and 70+ classes. Let's now analyses the ontological specification, based on the presented model, which was used to generate the application. In order to analyze only the specification part of the ontology, we exported the ontology in RDF/XML format and extracted only named individuals. From the resulting file we removed all blank lines and comments, so that each line represents either a named individual or a property. The ontological specification contains less than 2000 lines, 260+ instances and 1100+ attributes. The results show that we were able to generate more than 5 times more executable code than we specified using the ontological model. This means that we have successfully raised the development abstraction level. Also, if we were to develop a new application, with a similar set of features, we would be able to reuse the complete ontological model and even parts of the specification. Development efforts would be required only to support new features, which are not already present in the model.

The ontological model, source code of the code generators and the example application are available on this link: http://gpml.foi.hr/DanielStrmecki/.

VI. Conclusion

In this paper we presented an ontological model for specifying and automatically generating complete, business Web applications. In our approach we wanted to avoid the mix of generated and custom code, so all new features should be built in the existing model and full application generated on a single click. This approach enables possible future reuse of all developed features. In addition, we provided free, public access to the model and source code of the generators. The ontologically-enabled, generative programming approach presented here is especially suitable for developing software product families, with a significant set of common features. Since development and maintenance of the ontological model and code generators requires additional resource investments, this approach becomes feasible only if a significant amount of features are reused on multiple products/projects. As connections in the ontological specification are defined via object properties, even the same specifications can be reused between products. For example, a database entity for handling addresses, specified for one product, can be connected to another project instance simply by adding a new object property. In this way, we are able to reuse the model, code generators and the specification with minimal modeling effort. Development efforts are only need when introducing new features to the model or when debugging any of the currently available ones. Our approach reduces the number of repetitive programing tasks, enables development of software product families on high abstraction level and with high level of reusability.

Disclaimer

This paper publishes, in English, parts of the PhD thesis Framework for Business Web Application Families Development Using an Ontological Model and Source Code Generators by Daniel Strmečki, mentored by Ivan Magdalenić. The original work is published in Croatian, in accordance with

the PhD study program on Faculty of organization and informatics, University in Zagreb, Croatia. To download the original work in Croatian, please visit the Croatian Digital Dissertations Repository[1].

REFERENCES

[1] X. Nianfang, Y. Xiaohui, and L. Xinke, “Software Components Description Based on Ontology,” in 2010 Second International Conference on Computer Modeling and Simulation, 2010, vol. 4, pp. 423–426.

[2] E. Visser, “WebDSL: A Case Study in Domain-Specific Language Engineering,” Gener. Transform. Tech. Softw. Eng. II, vol. 5235, pp. 291–373, 2008.

[3] D. Strmečki, I. Magdalenić, and D. Radosević, “A Systematic Literature Review on the Application of Ontologies in Automatic Programming,” Int. J. Softw. Eng. Knowl. Eng., vol. 28, no. 5, pp. 559–591, May 2018.

[4] K. Czarnecki and U. W. Eisenecker, Generative programming: methods, tools, and applications. ACM Press/Addison-Wesley Publishing Co., 2000.

[5] I. Magdalenić, D. Radošević, and T. Orehovački, “Autogenerator: Generation and execution of programming code on demand,” Expert Syst. Appl., vol. 40, no. 8, pp. 2845–2857, 2013.

[6] A. J. Wiebe and C. W. Chan, “Ontology driven software engineering,” in 2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2012, pp. 1–4.

[7] H. Happel and S. Seedorf, “Applications of Ontologies in Software Engineering,” 2nd Int. Work. Semant. Web Enabled Softw. Eng. (SWESE 2006), pp. 1–14, 2006.

[8] C. Calero, F. Ruiz, and M. Piattini, Ontologies for Software Engineering and Software Technology. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[9] F. Bartolo Espiritu, A. Sanchez Lopez, and L. J. Calva Rosales, “Towards an improvement of software development process based on software architecture, model driven architecture and ontologies,” Int. Conf. Electron. Commun. Comput., pp. 118–126, 2014.

[10] D. Gašević, N. Kaviani, and M. Milanović, “Ontologies and Software Engineering,” in Handbook on Ontologies, no. January 2016, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 593–615.

[11] E. K. Karatas, B. Iyidir, and A. Birturk, “Ontology-Based Software Requirements Reuse: Case Study in Fire Control Software Product Line Domain,” in 2014 IEEE International Conference on Data Mining Workshop, 2014, pp. 832–839.

[12] A. S. Andreou and E. Papatheocharous, “Automatic Matching of Software Component Requirements using Semi-formal Specifications and a CBSE Ontology,” Eval. Nov. Approaches to Softw. Eng., 2015.

[13] S. Il Kim and H. S. Kim, “Ontology-based open API composition method for automatic mash-up service generation,” in 2016 International Conference on Information Networking (ICOIN), 2016, pp. 351–356.

[14] H. A. Duran-Limon, C. A. Garcia-Rios, F. E. Castillo-Barrera, and R. Capilla, “An Ontology-Based Product Architecture Derivation Approach,” IEEE Trans. Softw. Eng., vol. 41, no. 12, pp. 1153–1168, Dec. 2015.

[15] R. Sinha, C. Pang, G. S. Martinez, J. Kuronen, and V. Vyatkin, “Requirements-Aided Automatic Test Case Generation for Industrial Cyber-physical Systems,” Eng. Complex Comput. Syst. (ICECCS), 2015 20th Int. Conf., pp. 198–201, 2015.

[16] D. Toti and M. Rinelli, “Semi-automatic Generation of an Object-Oriented API Framework over Semantic Repositories,” in 2015 International Conference on Intelligent Networking and Collaborative Systems, 2015, pp. 446–449.

[17] M. Vanden Bossche, P. Ross, I. MacLarty, B. Van Nuffelen, and N. Pelov, “Ontology driven software engineering for real life applications,” Third Int’l Work. Semant. Web Enabled Softw. Eng, pp. 1–5, 2007.

[18] L. Tang et al., “WISE: A Prototype for Ontology Driven Development of Web Information Systems,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 3841 LNCS, no. 60473072, 2006, pp. 1163–1167.

[19] J. Solis, H. Pacheco, K. Najera, and H. Estrada, “A MDE Framework for semi-automatic development of web applications,” Model. 2013 - Proc. 1st Int. Conf. Model. Eng. Softw. Dev., pp. 241–246, 2013.

[1] https://dr.nsk.hr/en/islandora/object/foi%3A3579