Bull. Nov. Comp. Center, Comp. Science, 42 (2018), 15–21

FlexyFace — a portable solution for application-independent interface implementation

Alan Hoarau, Mathieu Lorho, Michael Ostapkevich,
Alice Pchelkina, Clement Petit

Abstract. In this paper, we propose an approach to constructing the user interface to be based on the ontological description. Its advantages are pondered. The main value using the ontology is the reduced complexity and development time as well as the increased quality of resulting implementation. The requirements to a system using such an approach are formed. An architecture that meets these requirements is proposed.

Keywords: ontology, user interface, software portability.

Introduction

The requirements and user's expectations of what a program should be like have currently increased in importance. It is no longer enough just to implement a required functionality. It has to be complemented with a compelling user interface of a descent usability. As a result, a considerable fraction of the developer's time and efforts is spent on constructing the user's interface and related issues. The complexity of the construction process can be decomposed to the complexity of an application project domain and the complexity of the GUI software platform. When starting a new application, the developer has to deal once again with both of these aspects. It is further aggravated by ever-changing interfaces of the underlying GUI support libraries. Some UI related codes written in Trolltech QT 2.x would require a certain renovation when migrating to QT 4.x/5.x. Thus, an application developer has to spend a considerable amount of time and other resources for aspects of the design that seem to be rather of a system level than just yielding himself completely to solving a problem in a particular application domain that confronts.

The main objective of the project is to facilitate the construction of the user's interface with the help of an application programmer with a bias towards the domain of the numeric simulation. The main idea is to decrease a part of complexity related to the GUI software platform. In order to achieve this objective, we propose the usage of ontologies as a means of a high-level description of an application problem domain and of the interfaces suitable for applications within this domain. The developer defines the essential notions of a problem domain and the aspects of their visualization in the form

of ontologies. Having a complete ontological description, the UI generator automatically produces the UI related fragments of a source code for one or another GUI platform. This makes it possible to handle the complexity of the GUI platform just once, when a generator for a particular platform has been implemented, and then this generator for a multitude of applications of a vast variety of problem domains is employed.

1. The overview of ontologies, their applications and tools

The conceptualization [1] is a description of a certain problem domain in the form of its key notions, their properties, limitations and their interrelations. When presented in a formal language, such a conceptualization becomes an ontology. The concept of ontologies although foreseen by scientist and visionary Ramon Llull centuries ago has practically emerged in the context of computer science in the 1980-s. Since then it has gained an importance in a number of essential applications, such as decision making in event-driven systems [2], microformat support, search systems, data mining and information extraction from unstructured and semi-structured heterogeneous data sources in Internet [3, 4], geospatial applications [5] as well as many others. The last decade or so is marked by the emerging efforts to bring ontologies into the user's interface related issues [6–9].

The success of ontologies can probably be explained by a decrease in design complexities and in the improvement of certain qualities of yielded solutions that they has brought to the developers as well as to the users. The ontologies make it possible to describe a knowledge at a very high level of abstraction. Such ontologies provide an ecosystem for independent developers to create, exchange and consume formalized conceptualizations of actually any problem or application domain. The solutions based upon ontologies are typically more flexible and adaptable to ever-changing requirements and other external circumstances. Particularly it is trivial to support a multi-domain, a multi-interface and multi-protocol ecosystems [10].

In addition to the graph description languages, such as RDF/RDFS, a number of languages entirely dealing with the description of ontologies were introduced, such as: OWL [11], DAML+OIL [12], F-Logic [13]. The tools for a visual construction of ontologies were designed as well. The most widespread among them are: Protégé [14], KMgen [15], Altova SemanticWorks [16], Top Braid Composed Free Edition [17]. Among less known but those with a powerful functionality, one can mention OntoGRID [18].

2. The formation of requirements

By now a number of UI ontological construction technologies of a varying degree of maturity has been already available [6–9, 19]. They bring some

essential values when employed for the UI construction. Particularly, the technologies in question hide some of the complexities of the underlying GUI platforms. As a result, they increase the portability of a produced application. They also create an ecosystem for the UI fragment reuse. Some fragments can migrate among applications of a very versatile nature.

Our particular objective is to provide a solution capable of handling a huge amount of data, which is typical of numerical and imitational simulations. Nowadays, most practically useful models aggregate a tremendous amount of data, which is impossible to store, not to mention to process, it in its entirety within a single computer. This implies that the data are scattered among many remote hosts or cluster nodes. Whenever something is about to be visualized, it has to be remotely generated and then transferred to a client host. Thus, in addition to conventional requirements for the UI ontological systems, the following ones have to be met:

1. A system must provide the functionality not only for the user interface generation, but also for the model data processing and transmission.
2. A system must normally operate even when it confronts the amount of data that exceed manifold of capacities of a local host RAM and storage devices.
3. A part of the system must be totally uncoupled from any underlying GUI platform in order to be transferred to a remote computing node without any user interface capabilities whatsoever.

No complete solutions, which were reviewed by the authors, support all these requirements to a full extent. Thus, a new one was proposed to meet all of them. The solution is provided in the form of a system that contains a number of utilities and libraries of functions.

3. The proposed architecture

The environment, in which the proposed system operates, is a distributed computing system. It consists of a user computer (called the client host) and a cluster for high performance computations (Figure 1). They are connected by a high bandwidth network channel. The cluster consists of one gate and many computational nodes. Only a gate is reachable from the client host. All the computational nodes are visible only from a gate. Thus, no direct communication between the client and computational parts of an application is possible.

A user does not code using programming languages. Instead, he creates an ontology, which can be divided into three principal parts (Figure 2):

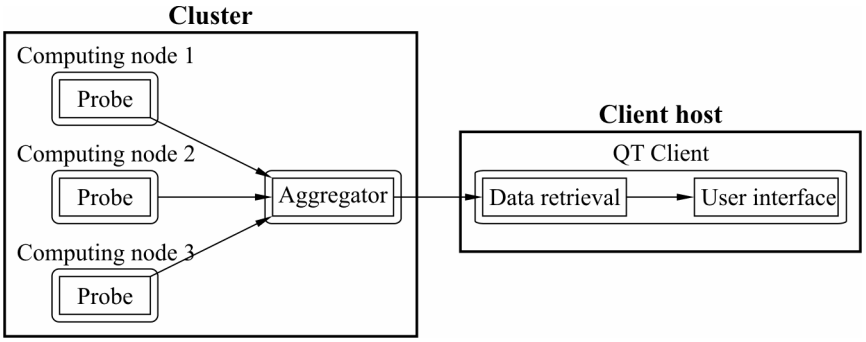

Figure 1. Distributed system environment

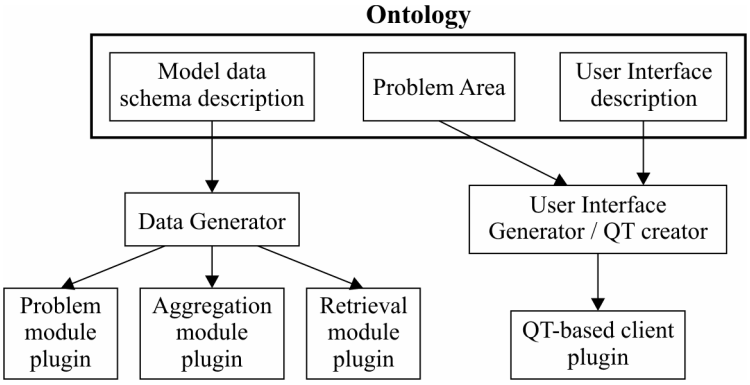

Figure 2. The data flow in a generator subsystem

1. The problem/application domain ontological description provides a repository of fundamental and essential types of entities, their properties, relations and limitations.
2. The user Interface description defines the modes of interface exploitation, scenarios of the user activities in order to accomplish all the required tasks and mapping of problem domain ontologies onto available elements of the underlying GUI platform.
3. The model data scheme description defines such aspects as data schemes and formats and the network protocols and interfaces to access the data and remotely invoke operations.

The system consists of the following main parts (see Figure 1):

The user interface consists of the ontological generator (see Figure 2) and the model interpreter. The former produces source fragments in a programming language (C, C++, Java, JavaScript depending on a particular GUI platform) as well as descriptions of windows, the menu and their compositions in one of the user interface description languages. The ontological

generator is used for static fragments of the application the user interface. The latter does not produce a compilable code. Instead, it interprets a high-level interface description at a runtime. The interpreter is suitable for dynamically changing parts of user interface. The source of dynamicity might be the conditions of environment, user preferences or model data flexibility.

The data processing plug in ontological generator builds fragments for functional interfaces, format parsing and all other sorts of issues related to data access in the distributed environment with an extensible set of systems to interoperate with.

The probe is a module for acquisition of data to visualize. The module is embedded into a user computation or a simulation program. It is executed at computing nodes in a parallel mode. Each parallel process of a distributed computational application has its own instance of acquisition module.

The aggregator is a tool that gathers all the data sent by several probes simultaneously and transmits the integral data to the data retrieval subsystem of the user client application. The aggregator is executed at the gate/control cluster node.

4. Conclusion

A prototype UI fragment generator is currently implemented for the Trolltech QT GUI library. Its visual construction interface is depicted in Figure 3. The OWL format is supported by the SPARQL library. In order to test the UI fragment generator, the ontology of university educational activity was created. A fragment of this ontology is depicted in Figure 4. The further

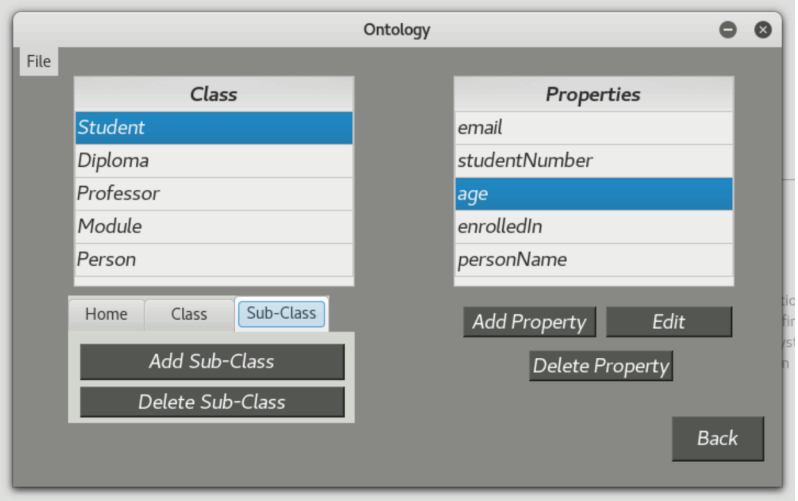

Figure 3. Snapshot of UI generator frontend

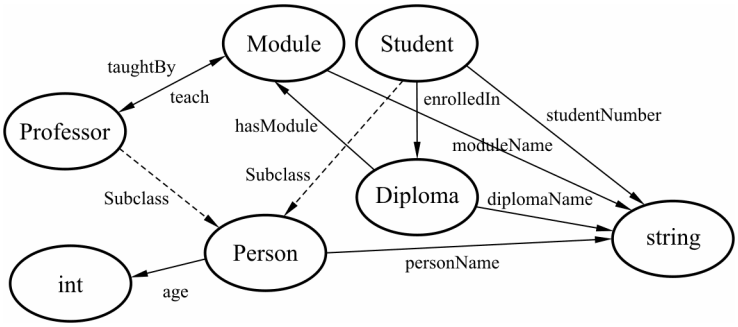

Figure 4. An example of problem-oriented ontology fragment

step to be taken is to implement the modules that would computationally support intensive distributed applications. The library is assumed to be used in a number of numeric applications and in a simulating system for algorithms and structures with a fine-grain parallelism.

References

[1] Guarino N., Oberle D., Staab S. What is an ontology? // Handbook on Ontologies. — Springer, 2009. — P. 11–17.

[2] Mansingh G., Rao L. Enhancing the decision making process: an ontology-based approach // CONF-IRM 2014 Proceedings. — https://aisel.aisnet.org/confirm2014/.

[3] Panov P., Soldatova L., Dzeroski S. Ontology of core data mining entities // Data Mining and Knowledge Discovery. — 2014.

[4] Buccella A., Cechich A., Brisaboa N.R. Ontology-based data integration methods: a framework for comparison // Revista Colombiana de Computacion. — 2005. — Vol. 6 (1). — P. 1-24. — https://doi.org/10.29375/25392115.1068.

[5] Ceh M. Universal Ontology of Geographic Space: Semantic Enrichment for Spatial Data / Marjan Ceh, ed. — IGI Global, 2012.

[6] Kleshchev A., Gribova V. From an ontology-oriented approach conception to user interface development // Information Theories and Applications. — 2003. — Vol. 10. — P. 87–93.

[7] Shahzad S.K. Ontology-based user interface development: user experience elements pattern // J. Universal Computer Science. — 2011. — Vol. 17, No. 7. — P. 1078–1088.

[8] Veselov A.V., Ostapkevich M.B., Piskunov S.V. An ontology-oriented approach to constructing user interfaces for an informational computational system to support innovations // Bull. Novosibirsk Comp. Center. Ser. Computer Science. — Novosibirsk, 2010. — Iss. 30. — P. 81–88.

[9] Piskunov S.V., Kratov S.V., Veselov A.V., Ostapkevich M.B. Using of assembly technology for the construction of users interfaces in network informational computational system // 2010 International Forum on Strategic Technology. — 2010. — P. 188–192.

[10] de Vergara J.E.L., Villagra V.A., Berrocal J. Semantic management: advantages of using ontology-based management information meta-model // Proc. HP Openview University Association Ninth Plenary Workshop (HPOVUA'2002), Boblingen, Germany. — 2002. — P. 11–13.

[11] OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition) W3C Recommendation 11 December 2012. — http://www.w3.org/2012/pdf/REC-owl2-syntax-20121211.pdf.

[12] Horrocks Ian. DAML+OIL: a description logic for the semantic web // J. IEEE Data Engineering Bulletin. — 2002. — Vol. 2002. — P. 4–9.

[13] Angele J., Kifer M., Lausen G. Ontologies in F-logic // Handbook on Ontologies. — Springer, 2009. — P. 45–70.

[14] Gasevic D., Djuric D., Devedzic V. Model Driven Engineering and Ontology Development. — Springer, 2009.

[15] Algorithm Information Technology Services. Reference Projects. — http://www.algo.be/ref-projects.htm.

[16] Altova Semantic Works 2012. User and Reference Manual. — Altova GMBH, 2012.

[17] Alatrich E. Comparison some of ontology editors // Management Information Systems. — 2013. — Vol. 8. — P. 18–24.

[18] Zagoruiko N.G., Gusev V.D., Zavertailov A.V., et al. OntoGRID System for automatic construction of subject domain ontologies // Optoelectronics, Instrumentation and Data Processing. — 2005. — Vol. 41 (5). — P. 11.

[19] Paulheim H., Probst F. Ontology-enhanced user interfaces: a 2nd edition survey // Int. J. Semantic Web and Information Systems Archive. — 2010. — Vol. 6, Iss. 2. — P. 36–59.