

Ontology as a backbone of future-proof software development

Patrik Kompuš¹

¹Faculty of Informatics and Statistics University of Economics, nám. W. Churchilla 1938/4, Prague, Czech Republic

Abstract

The world is experiencing a huge demand for software developers and, at the same time, the amount of skilled software developers is low. As a result, companies tend to hire unskilled resources. Therefore, the development of a system, in relation to business demands, is often very slow. Among various technical solutions to overcome this issue, separation of concerns between services is showing good results. I would like to show that we can apply the same logic in a not only technical solution and introduce (and demand) separation of concerns in the development itself. This would include correctly assigning ownership of the data model to data experts instead of development teams. The proposition is that ontologies are suitable means of supporting this separation. In this PhD project, I will attempt to partially verify whether a new methodology and approach based on ontologies will allow us to lower the number of resources spent on developing new products and future product features; these resources include time, personnel and costs.

Keywords

ontology modelling tools, semantics in services and processes, business optimization

1. Introduction


Software development has been a crucial part of almost every industry for the past decades, very often referred to as the "fourth industrial revolution". Not that it was not here even before, but since the hardware capabilities rose dramatically, the need for big amounts of software multiplied even more and we are long past the mark to get away without it. Be it in the world of data persistence (big-data management, cloud storage, etc.), libraries and other implementations of business logic on top of these data (micro-services, APIs, etc.), up to presentation towards human consciousness (explosion of front-end and UX pattern libraries, mobile apps, etc.). As a side effect, we are facing enormous demand for so-called "IT specialists" as a human resource. These roles accounted for 4.5 % of the total workforce in 2021 across the whole of the EU, an 1.3 percentage points increase from 2012. In numbers, the growth in that decade is more than 50 %, which was slightly less than 8 times as high as the corresponding increase (6.3 %) for total employment [1]. Every single company out there today has a version of an IT-department, e.g., there must be someone responsible for the company's IT infrastructure and daily operations. But let us focus on software development.

The 22nd International Semantic Web Conference, November 06–10, 2023, Athens, Greece

✉ qkomp00@vse.cz (P. Kompuš)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

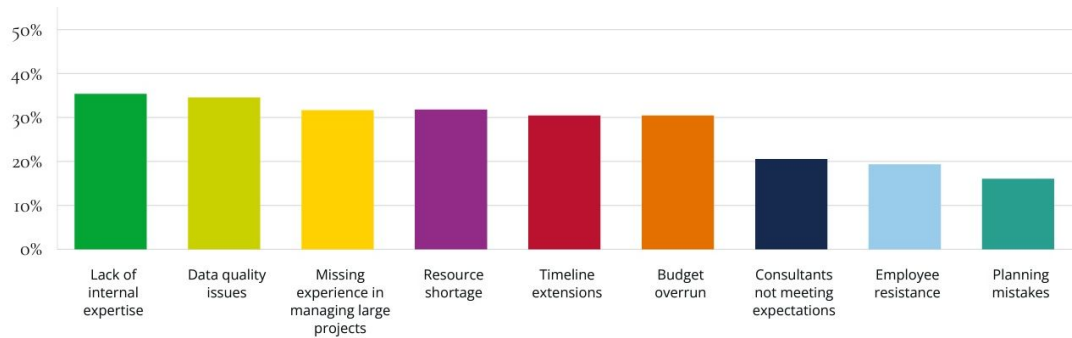


Figure 1: Natuvion 2022 Study - data and product transformation challenges

2. Problem definition

2.1. The problem

Since there is a huge demand for software developers, and the amount of skilled software developers is low, companies tend to hire unskilled resources. Their idea is following:

- invest time and resources into a junior developer,
- after few months the expertise would rise to mid-level, later to senior,
- such resources would then stay with the company for next decade, preferably longer.

Such flow generally works in non-IT related positions, but is hitting a wall very early in all IT fields. People flee to competition, as they offer them better compensation and benefits. This is nothing new, but what is new is the time they stay with one employer - the employee retention. It varies between 1.5 years to 3.2 years in big tech companies (Facebook, Tesla, Netflix, etc.) [2].

Looking as a human resources problem at first, the biggest challenge the companies are facing, when they lose an IT resource, is the knowledge they take with them. As a result, they must do the whole recruitment cycle all over again, and on top of that, they need to collect the knowledge of the leaving personnel and they need to do it very quickly.

The argument against might be that the knowledge is not really lost if it was documented, which is true. Unfortunately, because companies are hiring junior developers at a start, those often lack the inner urge to document what they have been working on. If they manage to hire mid-level or senior developers, they are often too busy to fix issues left by the gone junior developers or are busy reverse-engineering the code from gone senior developers, who took the knowledge with them. Therefore, the development is slow. When something is developed fast, due to pressure from stakeholders, errors are introduced.

A study by Natuvion from 2022 [3], where 200 companies responded to a structured survey which investigated experiences and challenges during their product and data transformation journey, shows the biggest challenge is the shortage of qualified personnel followed by data quality issues, see Figure 1.

In the same study, the companies were asked to define how quickly relevant changes to their systems have to be implemented in the future. About 95 % stated that changes should be introduced within a year, about 70 % indicated a time frame of six months, and 40 % even a period of only three months.

To summarize the problem:

Software in industries is very often developed slowly and/or with errors and it seems that one reason for it is huge demand for software developers. Companies hire unskilled resources and appoint them with core responsibilities. Instead of having access to the needed data knowledge or consulting with data experts, they tend to build data structures on their own, which causes mismatch between data experts' view and product owners' view over the same domain. As a result, developers are the owners of the data model. When the development team is assigned to another project, or simply leaves the company, the knowledge of data model is usually lost or incomplete.

2.2. Typically proposed solutions

This issue has been known for some time now and there are different angles of proposed solutions for it. Namely, these 3 are being used broadly:

- raise motivational incentives for developers, thus making them stay in the company longer (HR),
- outsource whole IT or some part of it (HR + Business management),
- introduce different methodology for software development, e.g. Agile and Scrum (Project management).

There is one more proposed solution, that, in my opinion, at least partly addresses the issue correctly. Micro-services (IT).

2.2.1. Micro-services

This is not a business, HR, or project management solution, but a technical solution. It decouples monolithic infrastructure into micro-services, where each service has one purpose only, and together they form a symbiotic ecosystem of services/components communicating through predefined channels and only across allowed layers of infrastructure. By doing that, each service has not just its own responsibility externally, but it also must assure its integrity internally. With such a requirement, the developer working on such service, is required to:

- communicate with external stakeholders (other developers) in a way, which starts to look more like a business relationship, rather than a friendly discussion. They are mutually agreeing on a contract between services;
- by working on a contract, they are in fact preparing the documentation for external parties trying to communicate with their service in the future. It is also used for any newcomer to work on that service, thus again serving externally and internally;

- as they are responsible for one service with one purpose only, they are keen on writing appropriate tests to keep the integrity of the service solid and not to be accountable if error occurs in the chain of instructions being invoked over several services/components. In other words, they do not want to be the one where the error is. Surprisingly, this comes to them naturally;
- as they work towards smaller objectives, they are putting high effort on re-usability of the code and therefore component itself. Again, this comes to them naturally.

For companies, moving towards micro-services, it requires huge investments in resources, but also time. Yet still, it is not the final solution for the problem of high demand for developers. Although it brought a great shift in the way how the developers perform, it did not go deep enough, e.g.: contracts are prepared, reviewed, and agreed between developers only and it is also only them who is creating documentation about things.

2.2.2. Research question

The problem summary and typically proposed solutions are subject of the main research questions:

Q1: Can non-technical separation of concerns in software development, meaning assigning the ownership of data models solely to data experts, help solving the technical problem of badly designed, not future-proof and unreliable software, as well as help solving the problem with knowledge loss within a business?

Q2: What methodologies and supporting tools can provide sufficient foundation for such separation of concerns and effective long-term dynamic persistence of domain knowledge?

3. Approach

Using **Ontology** as a foundation for data models and their use in development phase might solve the problem partially, if not fully. The idea of micro-services has shown how separation of concerns between services solves the technical issues. Using similar logical, not technical, approach of concern separation, I would like to show how it can be applied on a bigger scale and introduce and demand separation of concerns in the design and development itself. Once the knowledge is inside the graph, it is not bound to any expert nor technology, which solves the problem of crucial internal business knowledge and information loss.

This work is, not only, but also going to follow in steps of following projects:

- MOST - Marrying Ontology and Software Technology [4], a European research project with the goal to improve software engineering by leveraging ontology technology,
- ODSD - Ontology Driven Software Development [5], result of MOST and a way of moving from Model-Driven Software Development (MDS).

To answer the research questions, I would like to test my objective against real environment, meaning operating business. Partially, following goals are meant to be reached:

3.1. Methodology and supporting tools for data experts

Preliminary propositions are following: current data experts in the business (not researchers) often lack the knowledge of ontology, semantic technologies. Therefore, it is necessary to prepare, evaluate and later implement new methodology for data owners to easily manipulate with the models. Moreover, it needs to be backed up with lightweight but powerful toolkit/services-suite for them to be available to accomplish the modelling tasks without any technical obstructions.

This methodology should be strict on following:

- do not let developers construct the models about data that the service operates with. Let it be data owners who do this modelling and deliver it to developers as-is;
- do not let developers discuss the contracts between services but let the predefined data models rule the language throughout all communication channels;
- let data owners and product owners control the process of full quality assurance across services/components, based on the models defined by data owners, and requirements defined by product owners;
- let these models define the final contract to the outside world, thus having the possible external stakeholders discuss potential issues with data owners and not developers;
- let the whole knowledge of data structures and business requirements be outside of developers' domain and let it be documented and stored for anyone to access.

All mentioned above will be subject of broad field research, interviews with data experts and evaluation of the answers and information provided.

3.2. Methodology and supporting tools for developers

Just like new methodology is needed for data experts, it is also needed for the developers, who would be implementing the business requirements and will need to understand what the data experts have modelled. In fact, it will be more the services who would need to understand it, then the developers themselves. High level of automation and generalisation will need to take place and developers will need to start using predefined tests rather than verbal description of requirements.

3.3. Prototype of data pipeline software

As a use-case to demonstrate how the whole data-product collaboration can be backed with ontology and semantics, a prototype data pipeline to support the data models and data deliveries will be implemented. The idea is to create an event-based data supply pipeline, that would transform heterogeneous datasets into domain-specific homogeneous data events with their life-cycle and pass it on to the product as single small RDF graph, as opposed to big XML/JSON structures. Integrity of the whole data-delivery process, with emphasize on test driven development in data/mastering layer, shall be one of the cornerstones here and shall be described in detail. Proposed architecture see Figure 2. Technical implementation of such pipeline is not the key part of my research, it is rather the internal complications in communication and agreement between components of the pipeline and people developing and managing them. Knowledge graph should serve as mediator and source of truth for all the components' data models.

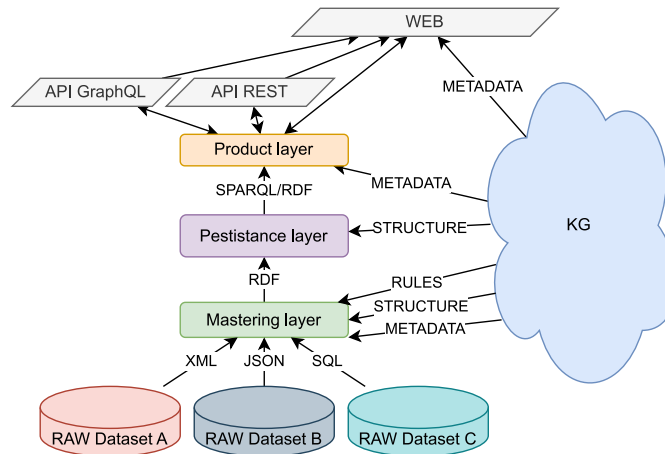


Figure 2: Data pipeline architecture

3.4. Prototype of product software

With data pipeline in place, prototype of product software would consume data events and prepare a persistent storage of the whole universe. Here, the methodology for developers would be put for evaluation. This too shall be implemented with test-driven approach, and in the same time, kept within the agile way of features implementation.

4. State of the art in research

The opportunity of using ontologies in software engineering projects has already been addressed by many. However, most of the previous approaches mainly focused on resolving particular technological problems, either related to the mapping between an ontological representation and the structure of implemented code, or way of developing the ontology itself.

Parreiras [6] discusses Model driven software development (MDS) in a synergy with Semantic web technologies, ontologies in particular, as a new way of enterprise computing. MDS is a way to form a contract between components based on a static UML model. Yet this model does not contain all relevant metadata, nor any representations of the modeled entities, like OWL does with it's individuals. Also the ownership of data models it is still within developers only. Using knowledge graph populated with ontology not only as a contract, but as a core part of the development dictating the structures anywhere in pipeline (and further at external consumers) allows for model and data content validation anywhere in any of the components, and in the same manner, thus reducing the implementation time. At the same time, the dynamic essence of ontology is kept and used, again throughout whole pipeline.

Ledvinka & Křemen [7] bring a very helpful comparison of existing mapping libraries between triples (semantic world) and objects (object-oriented programming). Yet this does not suffice the purpose of the research, the competence split between developers and data experts. Once object-oriented classes are prepared, developers are in charge, which drags them back to their

standard way of thinking.

Verborgh & Taelman [8] discuss and showcase how helpful for a developer's experience is, if they think outside of the box, for example with Linked Data abstraction. This is the view from the other end than than modelling - the application view. In my research, I would like to show how we can bridge these worlds of data experts and developers.

At ISWC in 2022, Hovland and Chrislock [9] described their need for sending small portions of data in non-traditional way to suffice their business requirements. Their proposal discusses implementation of so-called *versioned objects*, small RDF graphs. During the research I would like to study and evaluate this approach more in depth and possibly improve it with my findings while implementing the data pipeline prototype.

OWL-S: Semantic Markup for Web Services, a 2004 W3C Submission [10], is an ontology formally describing the web services. This is not essential for the proposed idea, but might be beneficial especially when integrating knowledge graph in product layer, therefore this will be further researched in my work.

Last but not least is the LOT project [11], which is oriented for engineering ontologies for usage in industries. The topic is very similar to the purpose of my research and also proposes methodology of ontology creation and maintenance, therefore it will be a major point in my evaluation. However, LOT methodology does not go further to the usage of ontologies and the data by further components in the chain, which is exactly what I would like to address in my research.

5. Evaluation

Main target of the research is to show how having the ontology as a core component in the whole system can speed up the process of implementation and augmentation of it, series of use-case scenarios will be selected from interviews with stakeholders. As for comparison, identical system/product will be developed using more standard technologies and tools, e.g. SQL, XML, JSON and no primary modelling language, as that is the situation in businesses nowadays - system A.

Once in place, performing the same use-cases with proposed methodologies and pipeline - system B, would visualize, whether using the ontology improved the implementation time, cut the resources and dependencies. The goal is to show, that the change request time-to-market is the key for business to be keen to adopt new methodologies. Therefore, the scenarios will mainly include structural changes, new product feature requirements, application of datasets usage restrictions and other compliance rules. Evaluation metrics will be defined based on importance level for the stakeholders, e.g. time-to-market, amount of resources needed etc.

Moreover, evaluations and comparisons with current state of the art tools will take place. LOT project, mentioned before, will be the closest competitor in terms of ontology creation methodology for data experts proposal, together with the tools being used.

6. Conclusion

This article proposes an idea and prototype of technical/architectural solution, using semantic technologies as a core component, for social and business problem of high demand for IT professionals, from the perspective of high employee retention and cost optimization, respectively. The idea is provocative and needs deeper research and evaluation. In case we will see significant time savings in the development phase, clear responsibility and concerns separation between data experts, developers and product owners, yet achieve the same, or even higher level of quality and integrity of demonstrated solutions, this will be a key achievement for majority of businesses, who are struggling with low-quality or late time-to-market solution and product delivers. It will also show that all stakeholders involved will need to change their view on their roles in the chain of solution and product development.

Acknowledgments

Here I would like to thank my supervisor, prof. Vojtěch Svátek, for his support and guidance.

References

- [1] Eurostat, Ict specialists in employment, 2022. URL: <https://ec.europa.eu/eurostat/statistics-explained/index.php>.
- [2] A. Levitsky, Facebook no longer has silicon valley's highest employee turnover, linkedin user data shows, 2020. URL: <https://www.bizjournals.com/sanjose/news/2020/12/30/employee-turnover-linkedin-data-2020.html>.
- [3] Natuvation, Transformation 2022, the study, 2022. URL: <https://www.natuvion.com/newsroom/challenges-2023/>.
- [4] Most: Marrying ontology and software technology, 2011. URL: <https://cordis.europa.eu/project/id/216691>.
- [5] J. Z. Pan, S. Staab, U. Amann, J. N. Ebert, Y. Zhao, *Ontology-Driven Software Development*, Springer Berlin Heidelberg, 2012.
- [6] F. Parreiras, *Semantic Web and Model-Driven Engineering*, Wiley-IEEE Press, 2012.
- [7] M. Ledvinka, P. Křemen, Comparison of object-triple mapping libraries, *Semantic Web Journal* 11 (2020) 483–524.
- [8] R. Verborogh, R. Taelman, Ldflex: A read/write linked data abstraction for front-end web developers, *ISWC 2 (2020)* 193–211.
- [9] D. Hovland, F. Chrislock, Versioned objects, *ISWC Industry Proceedings (2020)*.
- [10] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermot, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, K. Sycara, *Owl-s: Semantic markup for web services*, 2004. URL: <https://www.w3.org/Submission/OWL-S/>.
- [11] M. Poveda-Villalón, A. Fernández-Izquierdo, M. Fernández-López, R. García-Castro, Lot: An industrial oriented ontology engineering framework, 2022. URL: <https://doi.org/10.1016/j.engappai.2022.104755>.