On the design of enterprise ontology-driven software development

Summary

Situation and Goal

Due to factors such as hyper-competition, increasing expectations from customers, regulatory changes, and technological advancements, the conditions in which enterprises need to thrive become increasingly turbulent. As a result, the ability to change with an ever decreasing time-to-market, often referred to as 'agility', becomes an important determinant for the success of enterprises. As an enterprise and its supporting software can hardly be separated anymore, enterprise agility is to a large extent dependent on software agility.

Agility currently is mainly achieved by adopting processes that support flexibility with regard to planning and execution. In contrast to, e.g., a waterfall approach, agile approaches such as SCRUM and DevOps prescribe multidisciplinary teams that create new software versions in short iterations or even continuously. These approaches however say nothing about the structures in enterprises or its supporting software that should be adhered to in order to ensure quick adaptation is possible at all. It has been shown that changing software over time becomes more and more costly and indeed hampers enterprise agility.

The dream that drives this research is to have a structured approach towards software development for enterprises that supports the quick and continuous creation and adaptation of high quality software solutions to support enterprise agility. From the generic need for software quality, specific needs are identified, viz., that the software supports the end users, is evolvable, is created (or adapted) with little effort and in little time, and that the step from requirements to software constructs is traceable.

Partial answers are found in having a structured approach, i.e., a method. Methods for software development date back to the 1970s and mostly rely on the use of models. This is also known as Model-based Engineering (MBE), of which Model-driven Architecture (MDA), model-as-code and low code/no code are recent implementations. The reason that MBE and structured approaches gain new attention is the advancements in the availability of technologies as well as in (enterprise) modeling techniques. It is therefore that the challenge of this research is to work towards the creation of a new method for software development that answers the needs and supports enterprise agility. In order to meet this research goal, seven research questions have been formulated.

Approach

The dream that drives this research, is to have a method, including supporting tools, to create software from enterprise models that addresses the needs, can be applied repeatedly, and is adaptable to specific situations. In order to create such a method and supporting tools, clear specifications are needed. However, there are several uncertainties in order to create such a method, including what models to choose and whether such an approach is technically feasible.

It is because of these uncertainties that this research is considered part of the fuzzy front-end of creating a complete method. This fuzzy front-end is a necessary stage in which a problem is explored, guided by a vague idea of the solution, and assumptions are being examined, resulting in a minimum specification as well as an initial version of elements that could be part of the final solution. In this research the focus is on reducing these uncertainties, while aiming to find some initial method elements that could be part of such a method.

As methods often result from structuring or generalizing procedures or approaches that are being applied in practice, a practice-driven research approach is adopted. In order to deal with the uncertainties and answer the research questions, the Action Design Research (ADR) approach is adopted, combining Design Science Research with Action Research. ADR defines four stages of research that helps researchers to both make scientific contributions and to assist in solving current and anticipated problems of practitioners.

The research questions are answered initially through literature study, and validated or extended by practical research. The practical research consists of four exploratory case studies, aimed to explore the creation of a single method element. By combining theory and practice and by having multiple exploratory case studies, both rigor and relevance are added to this research.

Results and Benefits

The Model-driven Software Development (MDSD) approach is adopted as enabler for speed and traceability in the software development process. MDSD relies on model transformations for which both input models and target technologies are selected. From the possible enterprise modeling techniques, Design and Engineering Methodology for Organizations (DEMO) is chosen that provide the ontological enterprise models as the starting point for the MDSD approach. As DEMO models alone do not provide enough information to fully create working software, the Enterprise Implementation Framework has been developed. This framework uses Organization Implementation Variables (OIVs) to capture (additional) enterprise implementation design decisions, and can be used to consciously decide about the required flexibility on the enterprise level, that needs to be supported by the software. Together, DEMO and OIVs provide an answer to the other needs, i.e., completeness of user requirements and adaptability. An argument against this approach is that it moves complexity from the code to the (enterprise) models. As most complexity in software actually comes from the enterprise implementation, moving the complexity to these models is considered the only right approach.

Creating a (situational) method, mainly relies on the availability of so-called method fragments. Fragments can be categorized in three axes: perspective, abstraction and granularity. Existing literature is reviewed on the existence of method fragments that start from DEMO models and ends in some software implementation. While there are some usable fragments, none of these support all concepts from DEMO, most existing fragments ignore the step from DEMO to implementation model, and that there are almost no technical fragments to support the desired method.

For the four exploratory case studies four (modern) target technologies have been selected that all address multiple identified needs: microservices, mockups, Normalized Systems, and low code. The exploratory studies have in common that a mapping from the input, i.e., DEMO models and enterprise implementation (in terms of OIVs), to the target technology has been devised and evaluated in practice, sometimes on multiple enterprises (or enterprise models). These mappings, that are either automatable or automated during the exploratory case study, can be considered a method fragment, possibly to be assembled into an overall method. The exploratory case studies show that creating software from the chosen input models is indeed technically feasible, and that it is possible to do this in a structured (and automatable) way.

Put together, the foundation has been laid to create the desired method that addresses the needs and supports enterprise agility. The requirements are detailed and several fragments have been identified or created that can later be used to compose such a method. Such a method could be useful for practitioners as it may reduce software development costs and efforts, relies less on technically skilled people – so-called citizen development – and may improve project success rate.

Limitations and Future Research

The exploratory case studies were performed mainly in the Netherlands and mainly at public or semi-public organizations. Although there is no reason to believe that the results are not applicable to other enterprises, further validation is needed.

Moreover, most of the model transformations in the exploratory case studies involved manual steps. A reason simply is the unavailability of proper modeling tools for DEMO. In order to apply the algorithms in a disciplined way, it is necessary to further automate the mappings, and thus provide tool support to create the input models.

Embedding the suggested approach into existing approaches seems key to improve its adoption. Advantages are expected by introducing LLM and (generative) AI into the method. However, combining approach also means that they should be comparable, in their underlying theoretical background or ‘world view’, but also in the results they produce. Further research is needed in order to compare different approaches for software development from enterprise models.

Finally, adaptable software is just one factor in achieving enterprise agility. Other research shows that it is possible to apply principles to the enterprise level

that should enable enterprise agility. An open question is whether enterprise constructs can be created, perhaps based on one or more DEMO concepts, that inherently adhere to these principles. This also implies the need for an objective way to measure enterprise agility in order to be able to compare different enterprise constructs. Although the foundation was laid for creating enterprise software from enterprise models that intrinsically supports enterprise agility, there is still much to do in order to achieve true enterprise agility.