WASP Software Engineering and Cloud Computing
Assignment 2

Romuald Esdras Wandji
rewandji@cs.umu.se

October 14, 2022

1 Introduction

My field of research is based on Ontology-based data access (OBDA) and my project namely "Ontology-based update in VKG" concerns both foundational and applied research in the context of flexible and efficient management of large amounts of richly structured data, by relying on the paradigm of Virtual Knowledge Graphs (VKGs, also known as Ontology-based Data Access). Specifically, my Ph.D. project aims at introducing in the context of VKGs the notions of ontology-based update and evolution, and at studying foundational and applied issues related to this extension. The focus of research in VKGs has been on query answering, i.e., on using the ontology layer and the corresponding knowledge graph to extract data specified through a query from the underlying data sources. Instead, the problem of updating and/or dynamically evolving data sources through operations that are performed over the ontology (and hence the mapped knowledge graph) has not been considered so far. A solution to this problem would be of great practical relevance since it would allow the management of the key operations that are of interest in an information system (i.e., queries and updates) through the lens of an ontology.

2 Discussions

Software testing is one of the most crucial stages in any software development lifecycle and it's mainly done to make sure the software performs according to the required expectations. Automated testing is a set of procedures that are carried out by computers with the help of test scripts that are specified by human testers. Unlike manual testing which is performed by humans and at times is tedious, boring, repetitive, and most importantly error-prone, test automation can be used to automate repetitive tests or the ones that are difficult to perform manually. In the end, it automatically generates a report

comparing the expected result with the actual result outputted by the system under consideration, helping the teams gauge the quality of their system. The article published by the CISQ on "The Cost of Poor Software Quality in the U.S" states that poor-quality software costs around 2,84 trillion and that a $100 cost ing bug in the development stage can cost up to 10000$ in the production stage. Test automation can also provide benefits to various other aspects of software engineering, for instance, it offers faster feedback from testers, enables reusability of the written tests, and helps achieve continuous testing. Automated testing, despite its advantages however faces several challenges among which we can point out the lack of appropriate tool compatibility and interoperability, which is crucial for organizations with various technologies, applications, and platforms where the desire is to create a stable bridge between them. This challenge has been addressed by Nils Wild and al. in their paper "Test Automation Challenges for Application Landscape Frameworks". In my research, the need for automated testing is not crucial at the moment since, the only one ultimate test to be performed consists of checking whether the update semantic proposed consistently translates the ontology update into a database update. However automated testing may be considered at a larger scale of data integration for applications such as Ontop, Zapier, Boomi, and so forth.

Another important aspect of software engineering adjacent to automated testing is Software Maintenance and Evolution, it's in basic terms defined as a process of continually modifying and adapting software products after deployment in order to correct faults, and improve performance with the evolving environment so it matches both users and market requirements over time. It's generally used for large-scale projects and is known and characterized by its considerable cost and slow implementation speed. Improving the software's maintenance and evolution process remains an interesting open research topic. One solution based on empirical observation through the software evolution is the staged life-cycle model which mainly consists of five main stages namely: initial development, evolution, servicing, phaseout, and close down. Automated testing and maintenance in some sense aspect related to continuous deployment which is concerned with keeping user satisfaction by providing quick delivery in a continuous fashion. My research topic, however, is not concerned with this issue, though evolution in our framework might be of interest in the sense of keeping track of each change provided by the end users with the possibility of easily reverting or navigating to a specific state of the system.

The title 'ML will eat software' is almost considered a culture by ML engineers, and this at some point can be true and is largely due to the endless possibilities that can be implemented in ML. The period since 2010 saw unprecedented growth in the AI research community, Although several key ideas about AI have been around for a long, a number of processes have accelerated their potential use. Three main points can be pointed out to illustrate the impact of AI since its advent, first, the huge optimization of computer architectures and an increase in computer power, especially for AI chipsets, secondly the explosion

of data for training AI-related algorithm which extend its domain and decrease the cost of developing AI models.

3 Software Engineering future trends

It's with no doubt that AI has massively contributed to the improvement, automatization, and optimization of the core processes of several companies, for instance, in the transportation sector, it can be used to optimize transactions by reducing the number of idle transports. In the industry of software development, some tools powered by AI such as TabNine, TypeSQL, and BAYOU for generation and completion are created and available for use. Even in the sector of website development, some AI-based tools such as Microsoft AI lab are available to translate Sketch documents into Html, CSS, and js documents.

It's evident in the coming years, AI is likely to be more predominant in software systems as its application scope is getting wider. In my own opinion, with the knowledge I have in both AI and Software, I think Software and AI are kind of distinct in nature and that software will always exist. Therefore, AI will not eat software but will keep having considerable impacts on the software development process from the developers' perspective and will eventually change how we use the software.