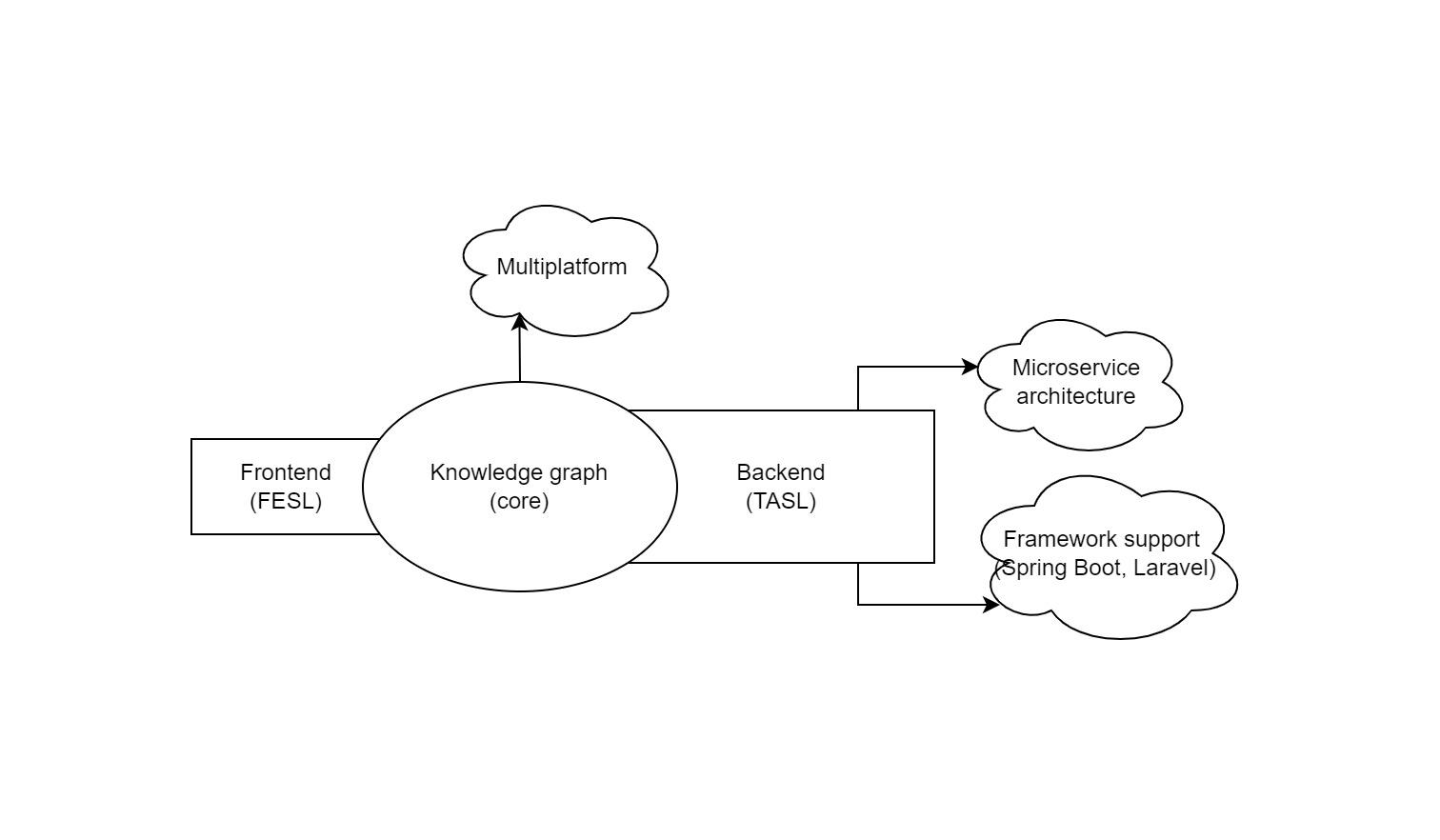
**RESEARCH PROPOSAL TEMPLATE**

|  |  |
| --- | --- |
| **Research title**  ***(Vietnamese)*** | Sinh ứng dụng microservice đa nền tảng từ biểu đồ tri thức |
| **Research title**  ***(English)*** | Generating Multi Platform Microservices app from Knowledge Graph |
| **Sub-committee** | IT |
| **Group name** | SweLab 2 |
| **Authors** | Ta Quang Tung  Nguyen Quang Huy  Phan Sy Tuan |
| **Supervisors** | Mr. Nguyen Van Cong (main supervisor)  Dr. Kazim Raza Talpur (co-supervisor) |

**Abstract**

* 1. Automating the software development process has long been a central topic in software engineering. The advent of modern AI methods and tools, especially the recent rapid advancement of generative AI has brought renewed research interests in the automation capability. A main limitation of the AI-based approaches is that they are only able to generate code fragments and not the complete software module. Our recent works have addressed the software generation problem for a popular type of backend application, named microservices application (MSA). Our works use embedded domain specific languages to express the MSA design specification and code generators to automatically generate complete MSA software modules. However, our works currently lack the ability to express the MSA specifications in higher-level languages, necessary to leverage the AI methods and tools. In this research, we aim to make a first step in bridging this gap by using knowledge graph to express the MSA specification, bringing it closer to the domain expert languages. We construct the knowledge graph model and implement it in a Java software framework, named JDA. We evaluate the effectiveness of the knowledge graph model against real-world MSA case studies.

**Keywords: domain-driven design, software generation, microservice architecture, knowledge graph**

1. **Introduction**
   1. 
   2. Fig. 1: Knowledge Graph based multiplat form fullstack code generation

In ResFes 2024, SWE lab proposed two parts of work.

* Team 1: focus on generating multi platform frontend application
* Team 2 (this team): focus on different features (architecture, database) for generating multi platform backend application

Expectedly, after finishing, we combine to create a Full stack KG model.

* 1. **1.1.Literature review**

Generative software development is not new in software development research. Many research papers discuss these topics, and some of them have proposed prototypes for generative software development tools.

Evans' seminal work on Domain-Driven Design (DDD) [[1]](https://www.zotero.org/google-docs/?broken=atj2eD) provides a foundational and comprehensive understanding of the principles and practices for building complex software systems from the core domain model. Czarnecki’s book on generative programming [[2]](https://www.zotero.org/google-docs/?broken=2UZyo6), on the other hand, offers insights into techniques for automating software development for complex applications. It introduces the concepts of code generation and metaprogramming, which are applicable to development automation for various types of software.

Fowler's book on domain-specific languages (DSLs) [[3]](https://www.zotero.org/google-docs/?broken=4uOm87) explores the benefits and techniques of creating languages tailored to specific problem domains. Among the applicable horizontal (a.k.a technical) domains include web application. Given the popularity of microservice applications [[4]](https://www.zotero.org/google-docs/?broken=VW8iyU) and the plethora of backend frameworks (e.g. SpringMVC, Laravel), there are significant benefits to be gained from applying DSLs to automate the development of the microservice apps.

Our recent works [7] [8] have proposed a method to automatically generate microservices applications for multiple backend frameworks. We applied a set of annotation-based DSLs (aDSLs) to express the software model and then proposed to develop code generator directly in OOPLs to parse the aDSL specifications and generate the backend microservice software. The aDSLs allow for the specification of the target backend framework (e.g. SpringMVC, PHP Laravel). We implemented our aDSLs and code generators in a Java software framework named JDA [[8], [10]](https://www.zotero.org/google-docs/?broken=KZ444o). The source code of this framework is freely available on Github at <https://github.com/jdomainapp/jda>.

However, a main limitation of our works is that it lacks a method to explicitly map the aDSL specification to the higher-level design models, such as those expressed in UML [[11]](https://www.zotero.org/google-docs/?broken=KtEwwH) or knowledge graph [[12]](https://www.zotero.org/google-docs/?broken=03i9Oh). This mapping, if expressed formally, can help leverage the benefits of our code generators for software development knowledge representation and, ultimately, for generative-AI capability.

Building this gen-AI capability requires a multi-phase research plan. Within the scope of this research, we are interested in bridging the first gap which is how knowledge graph can be used to formally express the aDSL specifications of the SPAs.

**Knowledge graph.**

A knowledge graph (KG) is a structured representation of knowledge, typically used to organize information in a way that's easily understandable for machines. It's essentially a graph database that connects entities, concepts, and their relationships. In a knowledge graph, entities are represented as nodes, and the relationships between them are represented as edges [[13]](https://www.zotero.org/google-docs/?broken=VTFMDQ).

Knowledge graphs are used in various applications. In Artificial Intelligent (AI), KGs facilitate tasks such as question answering, recommendation systems, and semantic search by organizing structured knowledge to enable efficient information retrieval and analysis, enhancing various aspects of artificial intelligence. [[14]](https://www.zotero.org/google-docs/?broken=085I1L)

By capturing and representing diverse types of knowledge and relationships, knowledge graphs enable more effective software development processes, Knowledge graphs have several applications in software engineering, aiding in various tasks throughout the software development lifecycle [[12]](https://www.zotero.org/google-docs/?broken=jWMGQx). For example:

1. Requirements Engineering: Knowledge graphs can be used to model and analyze requirements, capturing relationships between requirements, stakeholders, and system components. This facilitates better understanding of system requirements and helps in managing dependencies and traceability. [[15]](https://www.zotero.org/google-docs/?broken=Mnp2AX)
2. Software Design and Architecture: Knowledge graphs can represent software architecture, including components, interfaces, and their relationships. They help in visualizing and analyzing the architecture, identifying dependencies, and supporting design decisions. [[16]](https://www.zotero.org/google-docs/?broken=ev23sd)
3. Software Maintenance and Evolution: Knowledge graphs can support software maintenance and evolution by capturing information about changes, bug reports, and their relationships with code entities. This helps in understanding the impact of changes, managing software evolution.
   1. **1.2. The necessity of the research**

We **propose in this research** to bridge a gap in aDSL-based software generation for multiplatform **MSAs**, specifically with regards to formally expressing the aDSL specification in knowledge graph, with a goal to validate the validity of the specification.

This problem is formulated on the basis of our recent works in multi-platform **MSA** generation in OOPLs. Its resolution will help lay a foundation for our work in building a generative AI capability on top of our multiplatform code generators.

More specifically, we contend that this research will help bring the following benefits:

1. **Improved alignment with the domain model**: DDD emphasizes creating a rich domain model that accurately represents the problem domain. By generating **MSAs** directly from the knowledge graph, the architecture, databases would align better with the underlying domain model. This alignment helps promote consistency by minimizing the semantic mismatches between the frontend and backend.
2. **High-level multiplatform support**: knowledge graph operates at a higher-level of abstraction than aDSLs and so it is easier to incorporate into the graph the platform requirements while architecturing the software at the earlier phases of the software development life cycle. This increases the chance of the generated UI matching the user requirements for the target platforms.
3. **Novel application of knowledge graph**: Knowledge graphs provide a powerful means of representing complex domain knowledge and relationships. However, their potential in microservices application development remains largely unexplored. Our research would create a novel application for knowledge graph in generating **MSAs**.
4. **Advancing Software Engineering Practices**: The research contributes to the field of software engineering by combining concepts from DDD, graphs, generative programming, DSLs, and design patterns. By exploring the integration of these concepts, the proposed research advances software engineering practices and provides insights into novel techniques for microservices application generation.
   1. **1.3. Feasibility of research**

We adopted the feasiblity analysis approach by Dennis et al. [[17]](https://www.zotero.org/google-docs/?6Fe3vf) to analyse the feasibility of our research from two relevant perspectives:

1. Technical feasibility
2. Organisational feasiblity

The following table summarise the feasibility analysis.

|  |  |  |
| --- | --- | --- |
| **Feasibility factors** | **Feasibity measure**  **(Low, medium, high)** | **Description** |
| **Technical feasibility**: **Medium - High** | | |
| familarity with the problem domain | High | authors played an important role in implementing the multiplatform code generator in a previous work [[9]](https://www.zotero.org/google-docs/?womii7) |
| familarity with the technology | Medium | * authors are familar with the JDA framework and how to use it * authors are not yet familiar with knowledge graph (training will be provided by the supervisors) |
| project size (number of use features) | Medium-High | focus knowledge graph on the frontend part of the aDSL specification |
| compatibility with existing systems (JDA framework) | Medium-High | our project uses the JDA software framework as the base technology for code generation |
| **Organisational feasibility: High** | | |
| alignment with research theme of the software engineering (SWEng) lab | High | this research is to be conducted as part of the software development automation research theme of the SWEng lab of Swinburne Vietnam (Hanoi). As such, support will be provided by the academic staff working in the lab. |

1. **Research objectives**

To achieve the stated research aim, we formulate the following objectives:

1. Analyse the concept of knowledge graph and its application to software development automation
2. Design a knowledge graph model to represent the aDSL specification of MSA
3. Implement the knowledge model into the JDA software framework
4. Evaluate the effectiveness of the knowledge graph model using the same MSA case studies that were used in our recent works
5. Demonstrate the added-value of knowledge graph to MSA software construction by using it to automatically validate the MSA specification
6. **Research scope**

The scope of this research is focused on exploring the effectiveness of knowledge graphs in MSA application development automation. It emphasizes developing a prototype software solution that leverages knowledge graphs to improve the design specification of MSA.

1. **Approach and Method**

We adopt the system engineering approach and apply a combination of the following research methods:

1. background research
2. domain driven design (DDD)
3. model-driven software engineering

Initially, the research will focus on exploring the concept of knowledge graphs in software development. This will involve conducting a comprehensive background research to gain a deep understanding of knowledge graphs and their potential applications in the context of microservices application development. By analyzing existing research, case studies, and industry practices, the study will identify relevant techniques, best practices, and potential challenges associated with utilising knowledge graphs effectively in this domain.

Subsequently, a prototype of the knowledge graph model will be developed and incorporated into JDA. This prototype will serve as a proof-of-concept, showcasing the integration of knowledge graphs into the generative software development process.

To evaluate the effectiveness and feasibility of the prototype, a benchmark microservices application will be established. This benchmark application will serve as a reference point, representing a set of requirements and quality criteria that the generated applications should aim to meet. The benchmark will be carefully designed to encompass various aspects of microservices development, such as functionality, usability, performance, and maintainability.

Using the prototype, a microservices application will be generated based on the requirements specified by the benchmark. The generative process will utilise the knowledge graph and the features incorporated in the prototype to automatically generate the application code. The generated application will then be evaluated against the benchmark in terms of the effectiveness of the development process and the extent to which it meets the specified requirements and quality criteria.

The evaluation will involve quantitative and qualitative assessments. Quantitative metrics, such as development time, code quality, and performance benchmarks, will be used to measure the effectiveness and efficiency of the generative software development process. Qualitative evaluations will involve expert reviews, user feedback, and comparisons against industry standards to assess the extent to which the generated application meets the benchmark and satisfies the desired microservices development criteria.

The prototype and its evaluation against the benchmark will serve as evidence of the capabilities and to highlight the limitations of the knowledge-graph-based generative software development system and its potential practical applicability in real-world scenarios.

1. **Research plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Date** | **Task** | **Output** | **Person in charge** |
| **1** | **30/04/2024** | Analyse the concept of knowledge graph and its application to software development automation | Literature review with detailed analysis of research gaps | Ta Quang Tung  Nguyen Quang Huy  Phan Sy Tuan |
| **2** | **15/5/2024** | Design a knowledge graph model to represent the aDSL specification of MSA | Knowledge graph model | Ta Quang Tung  Nguyen Quang Huy  Phan Sy Tuan |
| **3** | **15/06/2023** | Implement the knowledge graph model into the JDA software framework | Knowledge graph model code in Java, integrated as a module in JDA | Ta Quang Tung  Nguyen Quang Huy  Phan Sy Tuan |

1. **Expected results / Dự kiến kết quả đề tài**

The expected result of this research is the development of a functional prototype of a generative MSA development system that incorporates knowledge graph to represent the MSA specification. The effectiveness of the prototype is evaluated.

We contend that the research makes a significant contribution to the automated software engineering research.is expected to demonstrate the effectiveness and feasibility of utilising knowledge graphs in microservices application development.

**References**

[1] E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston: Addison-Wesley Professional, 2004.

[2] K. Czarnecki, “Generative Programming: Methods, Techniques, and Applications Tutorial Abstract,” in Software Reuse: Methods, Techniques, and Tools, C. Gacek, Ed., Berlin, Heidelberg: Springer, 2002, pp. 351–352. doi: 10.1007/3-540-46020-9\_38.

[3] M. Fowler and T. White, Domain-Specific Languages. Addison-Wesley Professional, 2010. Accessed: Dec. 04, 2016. [Online]. Available: http://my.safaribooksonline.com/book/software-engineering-and-development/ide/9780132107549

[4] P. Klauzinski and J. Moore, Mastering JavaScript Single Page Application Development. Birmingham: Packt Publishing, Limited, 2016. [Online]. Available: https://learning.oreilly.com/library/view/mastering-javascript-single/9781785881640/

[5] Evan You, “Vue.js.” Accessed: Jul. 04, 2022. [Online]. Available: https://vuejs.org/guide/introduction.html

[6] Meta, “React.” Accessed: Jul. 04, 2022. [Online]. Available: https://react.dev/

[7] Google, “Angular Docs.” Accessed: Jul. 04, 2022. [Online]. Available: https://angular.io/docs

[8] D. M. Le, A. P. Nguyen, L. Q. Tran, H. T. Le, and H. V.-A. Tong, “Generating Multi-platform Single Page Applications: A Hierarchical Domain-Driven Design Approach,” in Proc. Int. Conf. 11th SOICT 2022, in SoICT ’22. Hanoi, Vietnam: ACM, 2022, pp. 344–351. doi: 10.1145/3568562.3568566.

[9] D. M. Le, C. V. Nguyen, L. Q. Tran, and K. M. Hoang, “Modularly Generating Multi-platform Single Page Applications with FrontEnd Patterns,” in Intelligent Systems and Networks, in LNCS. Singapore: Springer Nature, 2024, p. (To appear).

[10] D. M. Le, D.-H. Dang, and H. T. Vu, “jDomainApp: A Module-Based Domain-Driven Software Framework,” in Proc. 10th Int. Symp. on Information and Communication Technology (SOICT), New York, USA: ACM, 2019, pp. 399–406. doi: https://doi.org/10.1145/3368926.3369657.

[11] OMG, “Unified Modeling Language version 2.5.1,” OMG, formal/2015-03-01, 2017. Accessed: Sep. 29, 2020. [Online]. Available: https://www.omg.org/spec/UML/2.5.1/PDF

[12] L. Wang, C. Sun, C. Zhang, W. Nie, and K. Huang, “Application of knowledge graph in software engineering field: A systematic literature review,” Information and Software Technology, vol. 164, p. 107327, Dec. 2023, doi: 10.1016/j.infsof.2023.107327.

[13] D. Fensel et al., “Introduction: What Is a Knowledge Graph?,” in Knowledge Graphs: Methodology, Tools and Selected Use Cases, D. Fensel, U. Şimşek, K. Angele, E. Huaman, E. Kärle, O. Panasiuk, I. Toma, J. Umbrich, and A. Wahler, Eds., Cham: Springer International Publishing, 2020, pp. 1–10. doi: 10.1007/978-3-030-37439-6\_1.

[14] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, “A Survey on Knowledge Graphs: Representation, Acquisition, and Applications,” IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 2, pp. 494–514, Feb. 2022, doi: 10.1109/TNNLS.2021.3070843.

[15] “(4) (PDF) A Framework for Requirements Engineering in a Digital Integrated Environment (FREDIE).” Accessed: Apr. 21, 2024. [Online]. Available: https://www.researchgate.net/publication/242311156\_A\_Framework\_for\_Requirements\_Engineering\_in\_a\_Digital\_Integrated\_Environment\_FREDIE

[16] M. Ali Babar, T. Dingsøyr, P. Lago, and H. Vliet, Software Architecture Knowledge Management. 2009. doi: 10.1007/978-3-642-02374-3.

[17] A. Dennis, B. H. Wixom, and R. M. Roth, Systems Analysis and Design, 6th ed. Hoboken, NJ: Wiley, 2014.

**Appendices (*if any*)**