Available online at: https://jurnal-fkip.ut.ac.id/index.php/ijrse/issue/archive
International Journal of Research in STEM Education (IJRSE)
ISSN 2721-2904 (online)
Volume 6 Number 2 (2024): 14 - 23

Integrating the Software Development Lifecycle into Work-Integrated Learning: *A Case Study at a South African University of Technology*

[1]Mpho Mbele1, **[2]James Swart**
[1, 2]Central University of Technology
[1]mmbele@cut.ac.za, [2]aswart@cut.ac.za

Abstract

Work-integrated learning forms a vital part of student development as they obtain needed exposure to industry that can help them acquire relevant hands-on experience. Enabling students to acquire this type of learning may be achieved through actual placements in industry or through project-based learning at a higher educational institution. The purpose of this article is to evaluate the impact of integrating a Software Development Lifecycle process into work-integrated learning, specifically in software development projects at a South African university of technology. The methodology involved a comparative analysis of two student projects for the 2022 calendar year. Students in Project A used the waterfall model of the SDLC process, which may not have given them enough time for iterative review, which resulted in a substandard conversion between the class diagrams and route classes. Students in Project B used the agile model of the SDLC process, which encourages frequent reviews and permits essential modifications during the implementation phase, which resulted in alignment between the class diagram and route class. A further comparison between the two projects highlighted a key difference, being the presence (as in Project B) or absence (as in Project A) of clear directional indicators with comprehensive descriptions of the purpose of the relationships that should exist between the classes in the class diagram. It is recommended that students in information technology adopt an appropriate SDLC model that meets their specific project needs within a work-integrated learning environment.

Keywords: *Work-integrated learning, software development life cycle, comparative analysis, higher education, agile models, plan-driven models*

INTRODUCTION

"Our principal role as designers is to accelerate new ideas and the adoption of new ideas (Brainy Quote, 2023)". These words by a Swiss designer, Yves Behar, well illustrate the need to adopt new ideas, and especially in a design process. Design is fundamental to many fields of study, including engineering and Information Technology (IT).

It is necessary to remember that design exists in the mind foundations of all humans; in other words, all humans are designers at the "novice" level (Amraee et al., 2023), with progression to higher levels of expertise being acquired through learning and practice. This progress is essential in software development, which has been called "the backbone of today's digital economy" (Saeedi & Visvizi, 2021) . Best practices by academics are required to educate students in higher education to rise to a level of expertise that can be used to significantly contribute to this economy. On the other hand, students must also demonstrate through practice what they are capable of.

Integrating best practices into scientific software development requires allocating resources to make critical software design decisions and maintaining this focus throughout a project (Ruehl & Klise, 2021). An

Corresponding author:
Type Corresponding Name, email,
DOI: https://doi.org/10.33830/ijrse.v6i2.1675

example of a best practice is the use of the Software Development Life Cycle (SDLC) to enable students to practice and hone their skills to become the next generation of software developers.

The SDLC is a process that can be followed for the design and development of an application which includes plans describing how to design, develop, build, test, and deploy an application (Deepak & Swarnalatha, 2019). It has become an industry standard for software development. However, learning software development is a journey that takes a long time to complete (Loubser, 2021), and is challenging for some students in higher education. Adopting the SDLC into work-integrated learning (WIL) programs can mitigate this challenge and enable students to master software development skills.

WIL is a process where students spend a predefined number of hours working in an industrialized environment, gaining "hands-on" experience and "social skills." It represents a collaborative effort by industry and higher education to enhance student learning through facilitating the application of theory into real-life practice (Bates, 2011) and often embraces collaborative learning (Pearson & Daff, 2011). It has been suggested that WIL enhances skills, knowledge, competence, and experience that increase employability and leads to more satisfying and well-paid careers (Powell et al., 2008). Examples of appropriate undergraduate WIL in IT programs include internships, required professional practice, capstone projects developed with industry, and co-ops (Marinova & Momcheva, 2019).

This article evaluates the impact of integrating a SDLC process into work-integrated learning, specifically in software development projects at a University of Technology. The study's originality stems from comparing traditional plan-driven models with agile methodologies within an educational framework, which is a novel aspect. The methodology involved a comparative analysis of two student projects for the 2022 calendar year. The SDLC is firstly reviewed, followed by the context of the study. The research methodology is then given, followed by an analysis of two student projects that used the waterfall and agile models of SDLC.

Research Objectives:

This research aims to enhance the effectiveness of WIL in IT programs by examining the impact of adopting a SDLC process.

1. Assess the extent to which a SDLC process is followed in software development projects carried out in WIL at a University of Technology.
2. Analyze and compare the efficacy of the plan driven and agile SDLC methodologies by evaluating two student software development projects completed in the 2022 academic year, identifying the strengths and limitations of each approach.

LITERATURE REVIEW

The SDLC is a critical component of every software or system development project. It is a systematic approach to developing high-quality software at the lowest possible cost and in the shortest period (Merritt & Zhao, 2022). The SDLC defines the key phases of a life cycle that analysts, system designers, and developers use to plan and execute a series of actions necessary to generate the required system. Other models do exist, each with its own variants. The following are the typical phases in the classic SDLC, which are explained below (see Figure 1).

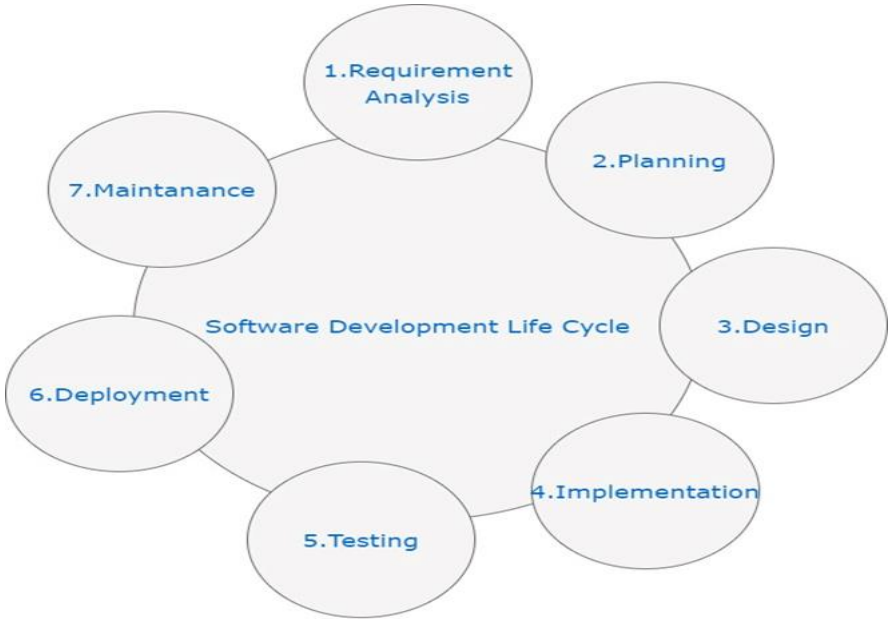

Figure 1: Software Development Life Cycle

The first phase is requirement analysis, whereby the development team collaborates with stakeholders to acquire and assess requirements. Once this is done, the next phase of planning follows: the project plan is produced to define the scope, timing, resources, risks, and budget for the project; this results in the Software Requirement Specification (SRS) document (Salve et al., 2018). The SRS is then used as input for the third phase of design to create the architecture of the software product under development. Initially, many architectures are built, and then the most effective one is chosen based on factors such as risk assessment, robustness, design adaptability, time, and cost (Shylesh, 2017). The design structure is translated into a programming language by the software developer in the fourth phase of implementation. Once the translation is complete, the result, in this instance, software, is handed over to phase five of testing. The testing phase must confirm that the program is error-free and produces the required results. After passing testing and meeting the requisite quality requirements, the program moves to the second last phase, called deployment, whereby it is deployed to a production environment for use. Configuring servers, databases, and other infrastructure may be required during deployment. Lastly, in the maintenance phase, the goal of software maintenance is to make the program function according to user requirements and to ensure service continuity (Navita, 2017).

To ensure the effectiveness of the software development process, several SDLC methodologies or models have been created to control the degree of complexity. There are two categories that may be used to characterize a set of models. One is plan-driven models, whereby the products are planned, and progress is calculated based on the plan (Choudhury & Nortjé, 2022). This approach is best suited for large teams and extremely critical products that are difficult to scale down. Secondly, there are agile models that represent a conceptual framework in the field of software engineering that involve a planning phase at the beginning, followed by a sequential progression towards the deployment phase (Al-Saqqa et al., 2020). Throughout the project's life cycle, there are iterative and incremental interactions. The primary objective of agile models is to minimize the administrative burden in the software development process by facilitating the incorporation of changes without jeopardizing the process or necessitating unnecessary rework. Agile models are particularly well-suited for small groups.

The focus is now directed to four noteworthy studies that investigated the usage of SDLC in different research contexts and fields using various models. The first two are plan-driven models, whereas the last two focus on agile models. (Herawati et al., 2021) focused on the design and implementation of a web-based Job Training Management Information System (JT-MIS) for Trunojoyo Madura University following the plan-driven

waterfall model as they saw that manually managing university practical work is time-consuming and error-prone. The results show that the implemented system effectively manages data related to practical work activities. The study also emphasizes the importance of integrating the JT-MIS with the university's academic system for further improvement. Overall, the JT-MIS streamlines practical work management and enhances efficiency.

Nugroho et al. (2018a) objective was to develop an Android-based marketplace application for agricultural products following the plan-driven prototype model with the aim of tackling the challenges experienced by farmers, improving the overall experience of consumers, and enhancing the agricultural marketing process in Indonesia. The authors demonstrated how the model was used for problem-solving through a structured process. The implementation of this model not only minimizes risks by identifying issues early on, but it also validates the model's adaptability to changing user needs, offering an actual demonstration of its importance in generating functional, user-responsive solutions.

(Suryantara & Andry, 2018)developed a medical record application in the health industry following an agile extreme programming (XP) model. Due to the patient data and health care demands, XP was selected because it is suitable for smaller teams working closely together and can quickly adapt to changes during application development. Their study focused on the value of carefully thought-out software development to guarantee quality and satisfy customer needs. The authors concluded that the successful application of the XP methodology provided a comprehensive and adaptable framework for the development of complex systems like medical record applications.

Žužek et al., (2020) investigated the implementation of agile project management practices in a Slovenian medium-sized company that specializes in wire harnesses for the automotive industry. Their goal was to demonstrate that, rather than adopting an entire structured agile project management methodology, non-software companies can implement a few practices and still benefit therefrom. According to the authors, this approach was chosen since traditional project management practices are no longer viable in today's dynamic and unpredictable project environment.

These four papers underline the need to select an SDLC methodology that correlates with the demands, dynamics, and objectives of a given project. It also demonstrates that SDLC can be used in a variety of sectors, including agriculture and health, and that it can be used to complete tasks in non-software companies. Another possible application is in education, which focuses on work-integrated learning, as discussed next.

CONTEXT OF THE STUDY

The study context is limited to a module called Work-Integrated Learning in Information Technology (WIL-IT). This is a compulsory third-year continuous assessment module within the Diploma in Information Technology (Software Development) offered at the Central University of Technology (CUT) in South Africa. This module is run in the second semester and has a duration of 13 weeks, which includes a comprehensive curriculum consisting of five main units. The conclusion of this three-year diploma provides students with 360 credits, 50 of which are attributed to WIL-IT (Central University of Technology, 2023).

The primary objective of this diploma program is to equip students with the knowledge, comprehension, competencies, and skills necessary for their progression toward becoming competent information technology (IT) professionals. The program enables graduates to be adequately prepared to undertake roles in various IT disciplines.

IT students are introduced to the fields of software engineering, mobile development, and web development in WIL-IT and are tasked with developing an integrated system (mobile and web applications) using two frameworks. Flutter is a cross-platform framework running on Android, iOS, and Fuschia that targets developing high-performance mobile applications (Tashildar et al., 2020a). ASP.NET is a framework for web development platform, which provides a programming model, a comprehensive software infrastructure, and

various services required to build up robust web applications for PCs as well as mobile devices (Komal, 2015). This integrated system must connect to a common online database that the students, in small groups of five to six members, have chosen. According to Davies, 2009a), small groups foster teamwork and collaboration by fostering communication, problem-solving abilities, problem-solving strategies, and individual attention. It simulates real-world situations, enhances project management communication, and allows for more individualized attention and active engagement, all of which improve the educational experience. According to (López-Fernández et al., 2019), small group sizes and active learning methods, such as project-based learning (PBL), have a positive impact on student motivation. These settings facilitate closer supervision by teachers, enhance student engagement, and contribute to a better learning attitude.

In the first semester of a calendar year, students must complete two modules: Software Engineering, which introduces students to SDLC, and Web Development, which introduces students to web application programming concepts. Once they enrol in WIL-IT in the second semester, students build upon this foundational knowledge and undergo a comprehensive application of the seven phases of the software development life cycle (SDLC) and how to effectively apply these phases for the development of mobile and web applications projects.

Assessment using three assignments.

The first assignment requires students to spend two weeks identifying a problem within an organization's existing software system and proposing a solution. They generate a system specification requirement document in accordance with phases one and two of the SDLC. This assignment contributes 20% to their final mark.

The second assignment requires students to spend three weeks on the creation of four distinct system designs: a use case, a class diagram, an entity relationship diagram, and a low-fidelity prototype, all of which correspond to phase three of the SDLC. This assignment contributes 30% to their final mark.

The last assignment requires eight weeks of work and focuses on the implementation of the proposed system, corresponding to phases four and five of the SDLC. For students to be assessed, they must present their work to the facilitators of WIL-IT, who use predefined rubrics to evaluate group performance and provide constructive criticism, improving students' learning and skill development throughout the course.This assignment contributes 50% to their final mark.

Once the final marks are calculated, the groups that have received 47-49% are given an opportunity to work on an extra assignment that assists the students to improve their marks.

Three facilitators, one skilled in software engineering, one in mobile development, and the third in web development coordinate the WIL-IT module. The number of groups assigned to each facilitator over the course of the 13 weeks depends on the number of registered students (on average each facilitator mentors ten groups). Once the groups are assigned, facilitators also take on the roles of mentors as well as simulated clients for the students' respective systems. Student groups with unique challenges are referred to relevant lecturers with the necessary skills within the IT department for consultations and assistance. Facilitators also ensure that students maintain a progress report for each assignment, fostering a comprehensive and hands-on learning experience. Furthermore, eThuto (the institution's learning management system built on BlackboardTM) also provides a wealth of educational materials to help students through each assignment, such as checklists, videos, and the rubrics to be used for assessing their projects.

RESEARCH METHOD

This research was conducted using a case study approach, which can be either exploratory, explanatory, or descriptive. Exploratory case studies, as indicated by Lucas et al. (2018) explore settings in which the case (intervention, for example) being evaluated has no clear or single set of outcomes. Zainal (2007) explains that explanatory case studies examine the data closely, both at a surface and deep level, to explain the phenomena in

the data, while descriptive case studies are set to describe the natural phenomena that occur within the data in question. The methodology was a comparative analysis of student projects from 2022, which enhances the understanding of the practical implications of different SDLC models in academic settings.

This study used an explanatory case study approach to analyse two projects qualitatively regarding their strengths and weaknesses based on class diagrams and code implementations across the two projects. This methodology will facilitate a deeper understanding of the underlying reasons why one project adhered to the prescribed design principles while the other project deviated from them, within the context of the SDLC. No ethical clearance was required, as no personal or private information was requested.

An analysis of projects submitted by two student groups.

Project A aimed to create an integrated system that serves as a preparation tool for job interviews for IT graduates. The system was designed to register graduates, and upon login, they could select a job category relevant to the role they might apply for, such as a software developer. Once a category was selected, graduates had the ability to view mock-up interview questions and potential answers. The system also included an administrator role that allowed for the uploading of these mock-up interviews based on job categories available in IT. The class diagram had eight distinct classes as shown in Figure 2 demonstrating an attempt to cover different facets of the system.

The interconnections among the eight classes in the diagram were ambiguous, causing confusion. Class diagrams are expected to represent diverse relationships with differing interpretations; this cannot simply be depicted with one specific arrow. Various shapes of arrows as well as descriptions of relationships should be clearly indicated. Each relationship should be accompanied by a clear explanation for comprehensive understanding. Moreover, only four of the original eight classes (A through D) from the class diagram were implemented in the route class, which is shown in Figure 3. The rest were modified, introducing errors and inconsistencies. This indicated a lack of adherence to the SDLC approach and caused a 'code-and-fix' scenario (unstructured approach for implementation), which deviated from the planned design and was time-consuming. Therefore, Project A was deemed non-compliant and unsuccessful.

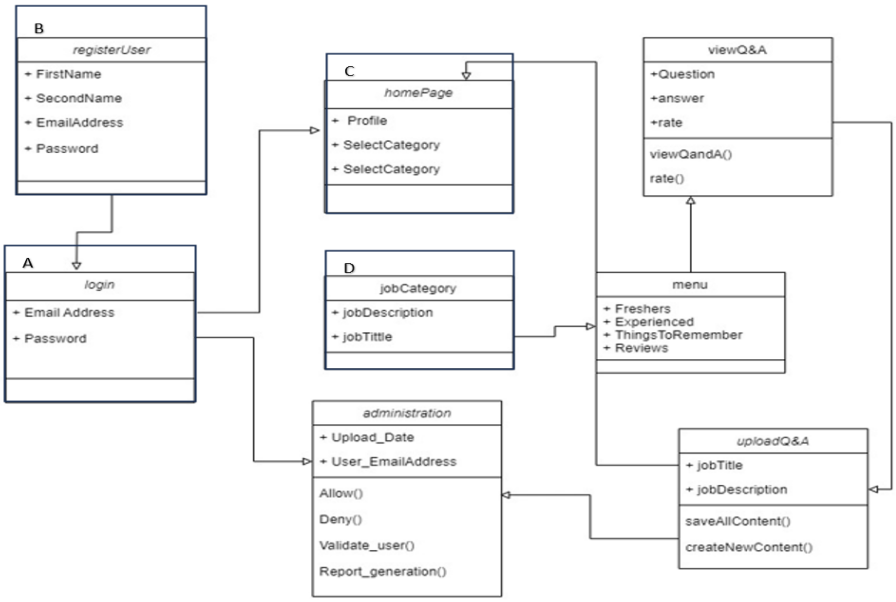

Figure 2: Project A class diagram

```
lib > routes > route_manager.dart > ...
19  class RouteManager {
20    static const String loadingPage = '/';
21    static const String loginPage = '/loginPage';  A
22    static const String registerPage = 'registerPage';  B
23    static const String bizhubHomePage = '/bizhubHomePage';  C
24    static const String profilePage = '/profilePage';
25    static const String editProfilePage = '/editProfilePage';
26    static const String serviceProviderPage = '/serviceProviderPage';
27    static const String serviceAddPage = '/serviceAddPage';
28    static const String categoriesPage = '/categoriesPage';  D
29    static const String viewServicePage = '/viewServicePage';
30    static const String viewRatingsPage = '/viewRatingsPage';
```

Figure 3: Project A route class

Project B aimed to create a voting application for students to elect their Student Representative Council (SRC). In this system, a student, acting as a voter, could cast a vote, edit their profile, and view voting results. The system also provided an administrator role with the capabilities to add, delete, or update candidates and to view and publish the voting results. The class diagram was effectively created, reflecting accurately the relationships among all seven classes with proper descriptions as seen in Figure 4.

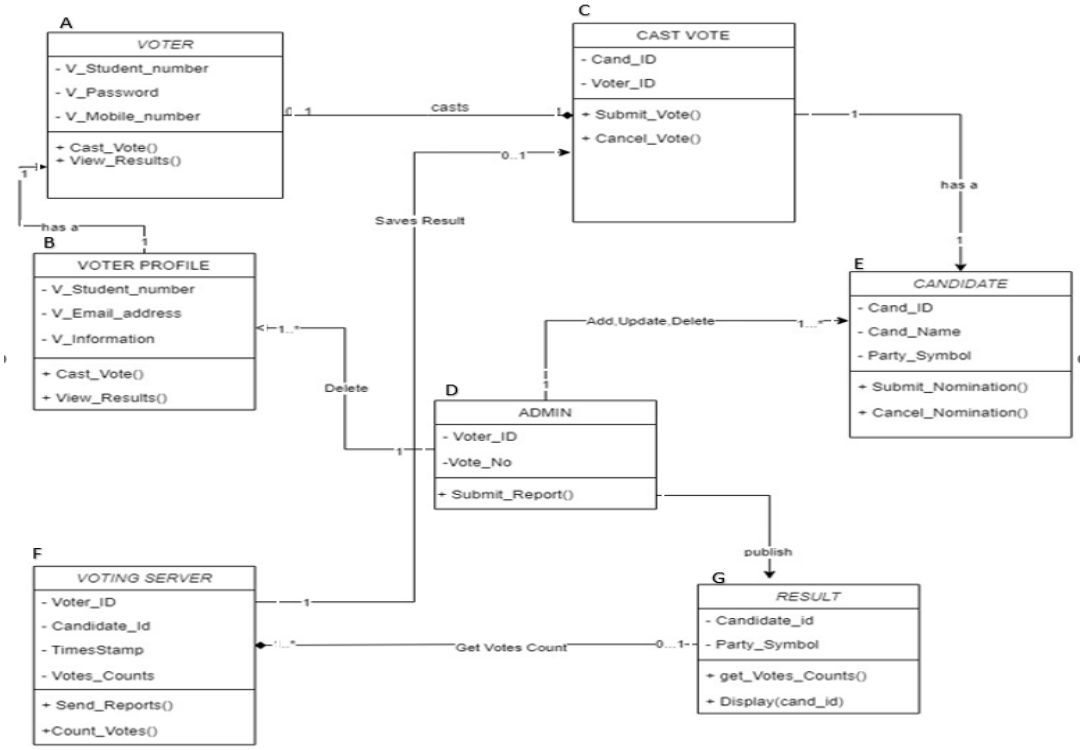

Figure 4: Project B class diagram

During implementation, the student group maintained the integrity of the original class diagram, ensuring that route class declarations and their relationships matched the original design. As shown in Figure 5, this is in accordance with the SDLC approach, enabling a smoother implementation phase.

All seven classes demonstrate the relationship between the class diagram and the route class. A potential weakness with this project includes adhering too strictly to the original class diagram, potentially limiting flexibility if requirements or circumstances change during the later stages of the project's development life cycle.

In these two examples, the implementation of SDLC significantly impacts the success of student projects. The class diagram for the non-compliant project (Project A) suffered from revisions and additions that were not part of the original design, resulting in confusion and inconsistency. In contrast, the class diagram of Project B replicated the original design for the project that conformed to the design, resulting in a more successful implementation in terms of design compliance. This shows that project B followed the SDLC by being able to convert the class diagram as it is to the route class.

Moreover, it is critical to indicate that students had the freedom to select an SDLC model that best suited the unique requirements and structure of their projects. Students in Project A used a waterfall model which was rigid and linear; there may not have been enough time for iterative review or error correction over the project's lifespan, which resulted in a substandard conversion of the class diagram to the route class. On the other hand, Students in Project B used the agile model which encourages frequent reviews and permits essential modifications during the implementation phase because it is iterative and flexible. Due to the students' ability to preserve the integrity of the original concept, the project was successfully completed.

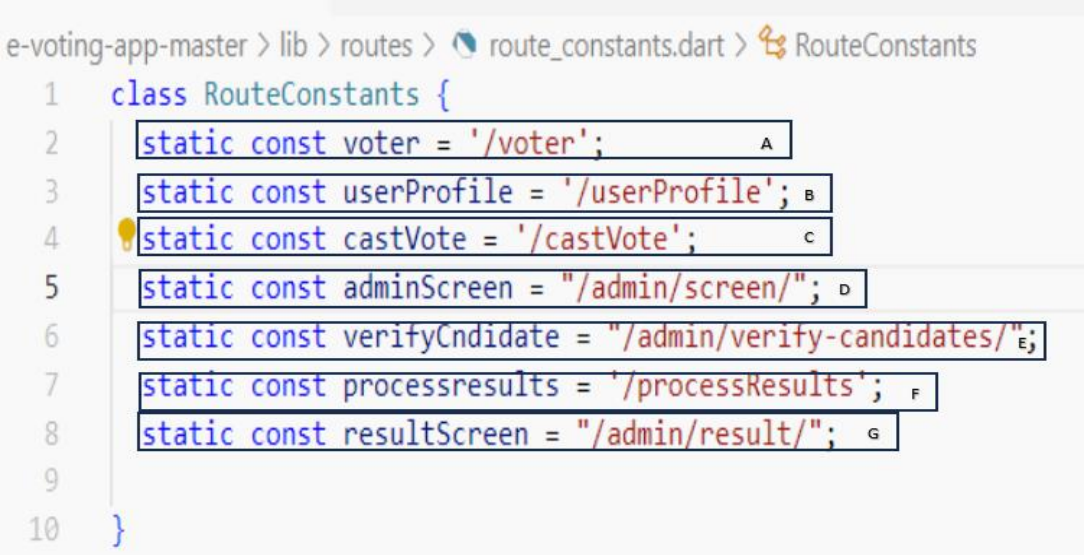

Figure 5: Project B route class

CONCLUSIONS

This article evaluated the impact of integrating a SDLC process into work-integrated learning, specifically in software development projects at a University of Technology. The methodology involved a comparative analysis of two 2022 case studies for the WIL-IT module.

The first case, Project A, did not align with the SDLC process, leading to confusion and a less successful outcome. In contrast, the second case, Project B, strictly adhered to the SDLC process and achieved positive results. A comparison between the two projects highlighted a key difference, being the presence (as in Project

B) or absence (as in Project A) of clear directional indicators with comprehensive descriptions of the purpose of the relationships that should exist between the classes in the class diagram. Another observation relates to the alignment between the class and route diagrams as shown for Project B, but which was not evident in Project A.

The study emphasizes the value of integrating the SDLC for successful software development projects, and especially within work-integrated learning programmes in IT. However, the significance of students selecting an appropriate SDLC model that meets their project needs is emphasized. The study recommends that iterative and flexible models, like the agile model, are beneficial in academic and work-integrated learning environments due to their ability to accommodate changing requirements. It is further recommended that students choose the right SDLC model based on their project-specific needs, which could enhance their learning and preparedness for the industry.

This has the potential to help improve the learning experience of students, as they are molded into competent IT professionals that may be well-equipped to handle a diversity of project demands in their future careers. It is a step towards bridging the industry-academia gap and fostering the next generation of skillful software developers.

REFERENCES

Al-Saqqa, S., Sawalha, S., & Abdelnabi, H. (2020). Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies*, *14*(11), 246–269; DOI: https://dx.doi.org/10.3991/ijim.v14i11.13269

Amraee, B., Medghalchi, L., & Dastani, Z. (2023). A Methodological Comparison of the Processes of Product Design and Architectural Design. *The Monthly Scientific Journal of Bagh-e Nazar*, *20*(120), 17–28; DOI: https://10.2023.358072.5252BAGH10.22034

Bates, M. (2011). Work-integrated learning workloads: The realities and responsibilities. *Asia-Pacific Journal of Cooperative Education*, *12*(2), 111–124.

Brainy Quote. (2023, July 14). *Homepage 2023*. Http://Www.Brainyquote.Com/Quotes. http://www.brainyquote.com/quotes

Central University of Technology. (2023, September 7). *CUT | Information Technology*. Information Technology. https://www.cut.ac.za/programmes/information-technology

Choudhury, D., & Nortjé, N. (2022). The Hidden Curriculum and Integrating Cure- and Care-Based Approaches to Medicine. *HEC Forum*, *34*(1).

Davies, W. M. (2009). Groupwork as a form of assessment: Common problems and recommended solutions. *Higher Education*, *58*(4).

Deepak, R. D., & Swarnalatha, P. (2019). Continuous Integration-Continuous Security-Continuous Deployment Pipeline Automation for Application Software (CI-CS-CD). *International Journal of Computer Science and Software Engineering (IJCSSE)*, *8*(10), 247–253.

Herawati, S., Negara, Y. D., Febriansyah, H. F., & Fatah, D. A. (2021). Application of the Waterfall Method on a Web-Based Job Training Management Information System at Trunojoyo University Madura. *E3S Web of Conferences,Ternate, Indonesia, October 27-28*, *328*, 04026.

Komal, C. (2015). ASP.NET AND ITS FRAMEWORK. *International Journal of Innovative Research in Technology*, *2*(6), 261–264.

López-Fernández, D., Tovar, E., Raya, L., Marzal, F., & Garcia, J. J. (2019). Motivation of computer science students at universities organized around small groups. *IEEE Global Engineering Education Conference, EDUCON, Dubai, United Arab Emirates, April 08-10*, 1120–1127.

Loubser, N. (2021). Software Engineering for Absolute Beginners: Your Guide to Creating Software Products. In *Software Engineering for Absolute Beginners: Your Guide to Creating Software Products*.

Lucas, P., Fleming, J., & Bhosale, J. (2018). The utility of case study as a methodology for work-integrated learning research. *International Journal of Work-Integrated Learning*, *19*(3), 215–222.

Marinova, R., & Momcheva, G. (2019). Survey of Information Technology Undergraduate Degree Programs in Canada. *2019 IEEE Canadian Conference of Electrical and Computer Engineering, CCECE ,Canada May 04-08*, 1–4.

Merritt, K., & Zhao, S. (2022). Software Design and Development of an Appointment Booking System: A Design Study. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, *420 LNICST*, 275–294.

Navita. (2017). A Study on Software Development Life Cycle & its Model. *International Journal of Engineering Research in Computer Science and Engineering* , *4*(9), 2320–2394.

Nugroho, H., Hendriyanto, R., & Tisamawi, K. (2018). Application for Marketplace Agricultural Product. *International Journal of Applied Information Technology*, *2*(02), 58–67.

Pearson, C., & Daff, S. (2011). Collaborative delivery of work-integrated learning to Indigenous Australians in a remote community. *Asia-Pacific Journal of Cooperative Education*, *12*(1994).

Powell, S., Tindal, I., & Millwood, R. (2008). Personalized learning and the Ultraversity experience. *Interactive Learning Environments*, *16*(1).

Ruehl, K., & Klise, K. (2021). Short-term results versus long-term impact: Applying software development best practices to scientific software (No. SAND2021-14621C). In *Scien* (Vol. 3). Nature Publishing Groups.

Saeedi, K., & Visvizi, A. (2021). Software development methodologies, HEIs, and the digital economy. *Education Sciences*, *11*(2).

Salve, S. M., Samreen, S. N., & Khatri-Valmik, N. (2018). A Comparative Study on Software Development Life Cycle Models. *International Research Journal of Engineering and Technology*, *5*(2), 696–700.

Shylesh, S. (2017). A Study of Software Development Life Cycle Process Models. *In National Conference on Reinventing Opportunities in Management, IT, and Social Sciences ,SIMS Mangalore, India, April 23-24,* , 534–541.

Suryantara, I. G. N., & Andry, J. F. (2018). Development of Medical Record With Extreme Programming SDLC. *International Journal of New Media Technology*, *V*(1), 47–53.

Tashildar, A., Shah, N., Gala, R., Giri, T., & Chavhan, P. (2020). Application development using flutter. *International Research Journal of Modernization in Engineering Technology and Science*, *2*(8), 1262–1266.

Zainal, Z. (2007). Case study as a research method. *Jurnal Kemanusiaan Bil*, *5*(1).

Žužek, T., Gosar, Ž., Kušar, J., & Berlec, T. (2020). Adopting agile project management practices in non-software SMEs: A case study of a slovenian medium-sized manufacturing company. *Sustainability*, *12*(21), 1–17.