PlantUMLGen: A tool for teaching Model Driven Development

Brian Pando, Jose Castillo
Universidad Nacional Agraria de la Selva
Tingo Maria, Peru
{brian.pando, joseo.castillo}@unas.edu.pe

Abstract — **The demand for professionals dedicated to software development continues to increase. The architecture analysis and detailed design are part of the key efforts for the correct development and continuity of the solution. The teaching of software engineering still has many challenges to solve. One of these challenges is the teaching of software design, since the efforts made during the course often end up in a dead document when moving on to the construction phase. This problem also occurs in the industry. The reviewed works show different approaches to solve this problem. The objective of this work is to propose a tool based on the model-driven approach to create class diagrams and transform them into the source code. Also, you can make changes to the diagram during construction stages and stay in sync with the code. The results show an encouraging experience with the use of the tool by students of the software design course during an academic semester.**

Keywords - MDD; Education in Software Engineering; Teach Software Design.

I. Introduction

Software development is one of the most demanded activities in the entire world [1]. The COVID-19 pandemic brought the future closer to the present, increasing the demand for software developers [2] and the agility to produce software [3]. The needs of users have also increased, as a consequence, development times and the complexity of what is developed [4].

Software companies should deliver running code as soon as possible [5]. Since the mid-90s the academy and the software industry were looking for methods that allow to lighten frequent activities [6]. Thus, focus efforts on the logic and complexity of the requirements [4], especially in agile environments [7]. One used alternative is model-driven development (MDD) [6].

Working with models is very useful for communication and understanding of the system [7]. The use of diagrams avoids confusion in the software analysis and design phases and UML class diagrams are one of the most used diagrams these days [8] ,[9]. Design tools like PlantUML can be used to improve the quality of the source code especially in novice developers [10].

The teaching of software engineering has many challenges to solve ([11],[12]), including software design and the permanence of these designs during the software life cycle [11]. But, sometimes these documents end up in dead documents that don't stay in sync with the source code [13]. This feeling is shared both in academia and in industry. However, it is necessary to seek strategies to reverse this feeling from the training stages of professionals.

The MDD approach has been used in different contexts ([14],[15],[16]). However, despite the benefits of this technique [17], one of the widely discussed problems is the culture of using these types of techniques [9]. More than 30 tools have been created to build UML diagrams [18]. But, there is a deviation between design and implementation that must be reduced [19]. The preferred ways to generate these models are the text-based ones such as PlantUML [20].

The reviewed papers show efforts to propose the use of model-driven approaches to understand and model the functionalities of the system ([21], [22], [5], [6]). As well as generating source code working from diagrams [23] and UML profiles ([24],[4]). There were also jobs that support the software maintenance process, creating models from the existing code ([21], [25]), especially for developers who are in the training stage or are newbies.

This work proposes a friendly tool that allows software design students, using a model-driven approach, to apply and encourage the use of this technique. The tool is tested with students during 2 terms developing an MVC web application, considering that the MVC pattern is one of the most widespread in the industry [4]. For now, this tool was created for Laravel Framework and unlike others doesn't depend of an IDE.

This paper is organized as follows. In Section 2 briefly introduce a background about Model Driven Engineering (MDE) and Model View Controller (MVC) pattern. The Section 3 describes related work. Section 4 shows the proposed tool called PlantUMLGen. Also, the method for getting the students appreciations. Section 5, results and discussion. Finally, conclusions and future work.

II. Background

A. Model Driven Engineering and Development

Model Driven Engineering (MDE) is on top of the abstraction of Model Driven approach. MDE is a software engineering approach that considers models not only as documentation artifacts, but can also be used in all engineering disciplines and in any application domain [26]. The model driven engineering (MDE) is a technique widely used for

2022 17th Iberian Conference on Information Systems and Technologies (CISTI)
22 – 25 June 2022, Madrid, Spain
ISBN: 978-989-33-3436-2

developing web application, overall MVC projects [4]. In this way, models transform to executable systems.

Model Driven Development (MDD) approaches tend to define modeling languages to specify the system under study at different levels of abstraction, to provide transformations in order to improve the productivity and quality of the process and the final software system [26]. MDD aims to automate many of the complex (but routine) programming tasks, such as providing support for system persistence, interoperability, and distribution, which still have to be done manually [27].

Integration and interoperability are the main goals of MDD and the use of models is the technical way to achieve these goals. MDD goes through different levels of abstraction, from the business model itself to the final executable code [28]. Also, the motivation of MDD is to improve productivity by increasing the performance that a company gets from its software development effort [27]. Class diagrams and activity diagrams are the most used input notations in the MDD approach, while programming languages are used for output notations [28].

B. *MVC Pattern*

The MVC design pattern is made up of model, view, and controller. As a design pattern, MVC often breaks an application into separate layers that can be analyzed and sometimes deployed separately. By decoupling models and views, MVC helps reduce architectural design complexity, increase code flexibility and reuse [29], and reduce development and maintenance workload, as well as get a foundation solid for future updates [30].

In the MVC pattern, the Model is made up of application data and business rules, it is the core of the application and controls the access to the data and the updating of the data; The View is in charge of expressing the content of the model and receives data from the model and decides how to present the data. When the model changes, it will notify the view to maintain data consistency; The controller is in charge of coordinating the actions and events that affect the model and the view, it accepts customer requests and assigns them to the model implementation actions, and returns the result to the view [31]. This process is shown in Fig. 1.

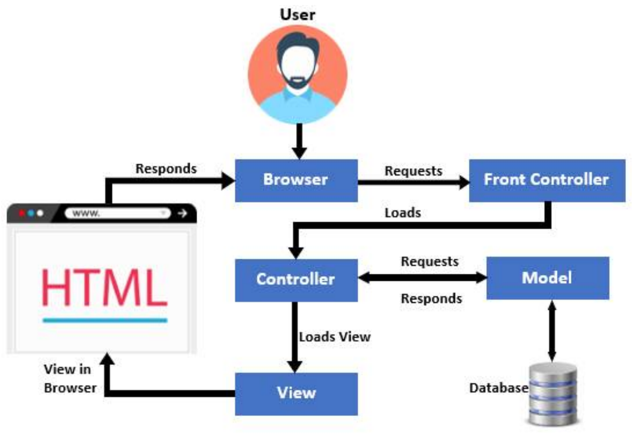


Figure 1. Model View Controller pattern [32].

MVC is suitable for the development of large scalable web applications, which separate the input, processing and output of the application into three central parts of the model, view and controller, so that they can perform their functions in such a way that any change does not affect to the other two parts [11]; and it has become the architecture based on many PHP frameworks [32] such as Laravel, Symfony, CodeIgniter, CakePHP and others.

The advantages of using the MVC pattern are: i) standard, consistency and predictability, ii) software components or building blocks that developers can share and reuse code, iii) a standard model or architecture that allows easy visualization of how the whole system works. iv) reusable code and extensively tested in libraries, classes and functions, v) well-structured code using architectural pattern, and vi) security, interoperability and maintenance [33].

The MVC pattern is still valid, so much so that [34] found three variants of the MVC architectural pattern: mixed view and controller; supervisor controller with relaxed view layer and supervisor controller with strict view layer.

III. Related Work

Madanayake [8] says than User Stories are a primary, important and high-level definition of the requirements. He proposes an approach using PlantUml to transform User Stories to Use Case Models. Finally, the work suggests than using this kind of approach is possible to generate more diagrams. Fabian [22] proposes an approach to create Use Case scenarios from the User Story. Using Natural Language Processing, he detect ambiguous stories and show relationship elements.

Morikawa [5] says the scenario and sequence diagram should synchronize for developing successfully. They propose a method to transform and keep synchronized the scenario diagram to sequence diagram using a language called SCEL for the scenario diagram and PlantUml for the sequence diagram. The proposal is used in some scenarios: (i) hotel reservations, (ii) train ticket and (iii) a library with successful results, but showing some problems concluding that sometimes the scenarios should be correctly written, also, sometimes the scenario diagram may be slightly modified for getting a correct sequence diagram.

Mythily [6] proposes an approach using model driven architecture (MDA) and logical prediction. It transforms the sequence diagram to class and activity diagrams, thus, save time and costs for modeling activities and guide the software development lifecycle. Using an illustrative example, show how to transform the diagram in a easy way.

Hernandez-Mendez [23] uses a model driven approach to consume RESTFul APIs for a single page application. He uses a query service, GraphQL and other components, to transform the model to NodeJS code. Then, developer can write the internal code, but the objects and methods are generated. The work shows an example using the proposal connecting with 2 services. The results present time reducing and well-structured code.

Wutthichai [24] proposes a tool for generate code for an single page application based on AngularJs using an UML profile. The tool is based on Enterprise Architect, StartUML and MagicDraw. The author uses the tool in a study case demonstrating that 87% of code is generated by the tool, thus, reducing time to coding and risk for bugs.

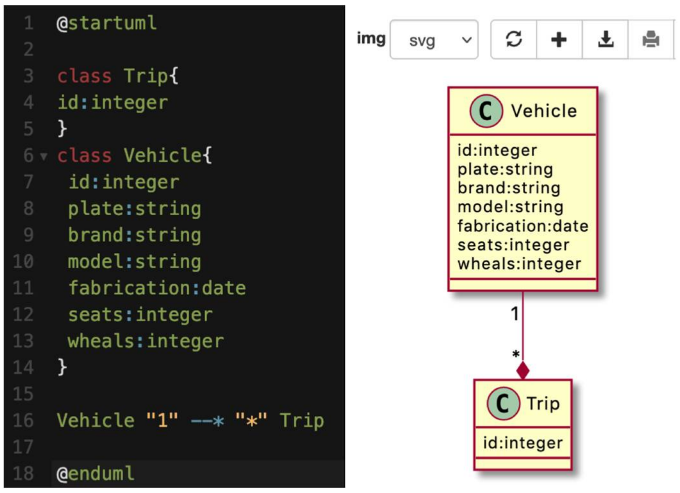
Ahmad [4] proposes a MVC model driven framework to transform UML diagrams to MVC frameworks like CodeIgniter, Struts and ASP.Net. They use a UML profile in Eclipse Papyrus and validate the proposal with two study case: (i) a JAVA file upload system and (ii) a PHP address book. The work demonstrates the improvement about the number of LoC between the existing code and the generated by the tool.

Some works use the a tool for doing reverse engineering. Yang [21] suggests using diagrams is easy to understand the code and its flow execution, overall, when the users didn't write the code. So, he proposes a tool for visualizing objects, classes and sequence diagrams using reverse engineering in a case study for learning Java. His results show a strong agreement using the tool.

IV. PLANTUMLGEN: THE PROPOSED TOOL

In MDD approach [35] there are 4 models and transformation between these models: CIM (Computation Independent Model), PIM (Platform Independent Model), PSM (Platform Specific Model) and source code. This study begins since PIM model, which works with UML[36][35] diagrams, and transforms to source code through proposed tool.

This tool is based on PlantUML, the open-source tool for writting quickly diagrams like: classes, sequence, use case, activities, deployments and others, using a simple textual language. It has extensions for some know IDEs like Eclipse and Visual Studio Code, and was used in several proposals to support agile software process [7]. Also, there are some tools online like PlantText or PlantUML Editor.



```
@startuml

class Trip{
id:integer
}
class Vehicle{
 id:integer
 plate:string
 brand:string
 model:string
 fabrication:date
 seats:integer
 wheals:integer
}

Vehicle "1" --* "*" Trip

@enduml
```

Figure 2. PlantUML for creating classes and relationships.

Left part of Fig. 2 shows the syntax of how to write classes in a textual language like PlantUML, then this code automatically is represented by a diagram showed in the right part of Fig. 2. PlantUML can create others diagrams, however is not necessary, because this work only needs the class diagram.

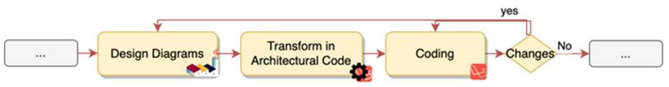
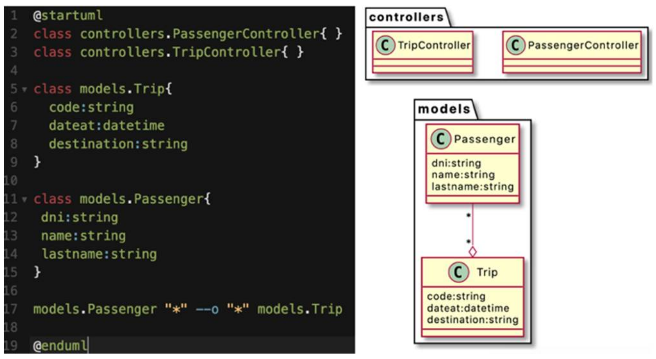


Figure 3. Proposed tool during the design and coding phase.

After the requirements analysis, continue the design process, until the code implementation. After that, if there are changes in design, the code will change too, like is shown in Fig. 3. The proposed tool allows for users analyze and create diagrams using PlantUml tool, specifically the class diagram, which will be transformed to source code for a Laravel prototyping project according the diagram. Fig. 4 shows a prototyping example, using plantUML.

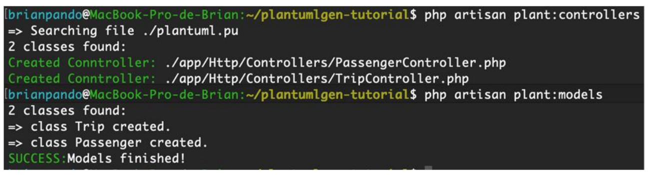
The tool uses PHP code to read the plain class diagram file script in PlantUML syntax like showed in Fig. 4. Thus, to find classes, properties, data types and relationships for creating class files with the Laravel syntax and to locate in folders according to a Laravel Project.



```
@startuml
class controllers.PassengerController{ }
class controllers.TripController{ }

class models.Trip{
  code:string
  dateat:datetime
  destination:string
}

class models.Passenger{
 dni:string
 name:string
 lastname:string
}

models.Passenger "*" --o "*" models.Trip

@enduml
```

Figure 4. Class diagram for MVC Laravel project.

To use the proposed tool, the users install the tool in their Laravel project using composer commands explained in https://github.com/brianpando/plantUmlGen. The class diagram script must be in the root project. Then, for creating controllers and models, this tool takes advantage of the "artisan" commands. The user only executes 2 commands: "php artisan plant:controllers" and "php artisan plant:models". The first one, create the controller's class and add some common methods like list, create, save, update and destroy a resource. The second one creates the Laravel models and the migrations for creating tables in database and keep synchronized. The Proposed tool is based on MVC architecture like Fig. 2. The users should write a class diagram using controller and model classes, packaging in namespaces according to Laravel convention. Currently it supports only composition and aggregation relationships. It is important to use tags in the relationship as a cardinality description. In this way, the tool uses the "artisan" commands to generate Laravel code. Fig. 5 shows an example results applying commands using the proposed tool.



```
brianpando@MacBook-Pro-de-Brian:~/plantumlgen-tutorial$ php artisan plant:controllers
=> Searching file ./plantuml.pu
2 classes found:
Created Conntroller: ./app/Http/Controllers/PassengerController.php
Created Conntroller: ./app/Http/Controllers/TripController.php
brianpando@MacBook-Pro-de-Brian:~/plantumlgen-tutorial$ php artisan plant:models
2 classes found:
=> class Trip created.
=> class Passenger created.
SUCCESS:Models finished!
```

Figure 5. Example for controllers and models using the proposed tool.

After the users execute both commands, they can code and implement every method to get projects up and running. Of course, during the development, the users add or remove methods and classes. The users must modify the class diagram and execute the commands again, because these commands check if the class exists and modify it adding methods or properties, in this way keep living the diagram and code.

This tool was created after some experience teaching topics like software architecture and software design. The students learn about the most popular architectural models and when could apply each one. The students analyze the software architecture using 4+1 technique and think how to transform part of requirements using this technique using diagrams like a class diagram. Then, the students must coding a software prototype using this diagrams. Finally, the course analyzes what happen when the software is on production and what happen if some requirements change and this change affects the designed software.

V. Methodology

The study was applied in a design software course. This course usually take 4 moths each year. This is a course in the 3rd year of the software engineering career, after courses like intro to programming, algorithms and requirements. So, the profile of participants are students of 5th semester with some knowledge about Laravel Framework for PHP.

The study had 2 iterations: in 2020, and 2021 course. During 3 months the students learned about concepts and steps for software design and some of Laravel in practice. The last month the student started learning MDD concepts. Then continue with cases using the tool. Table 1 shows the schedule for this last month.

In 1st session, the professor and students analyze an exercise and create the MVC class diagram. Then, in pairs, the students code in Laravel and explain the running prototype software in class. In 2nd session, the professor and students analyze the same exercise and create the MVC class diagram. But, the students use the proposed tool and code in Laravel. Then, they explain the running prototype software in class.

In 3rd and 4th session, the study has the same interaction like 1st and 2nd, but using a more difficult and challenging exercise. During this period, the students practice and develop a software prototype (final project). They demonstrate all capabilities learned during the course and they can use the tool for being quickly if they prefer.

TABLE I. Schedule for the last month of the course

Week	Session	Description	Session Time	Coding Time
1	01 02	Intro to MDD. Look for cases about MDD.	4 hrs	--
2	03 04	Case 1. Without proposed tool. Case 1. With proposed tool.	4 hrs	69 min 27 min
3	05 06	Case 2. Without proposed tool. Case 2. With proposed tool.	4 hrs	54 min. 21 min.
4	07, 08	Survey and course final evalution.	4 hrs	--

The generated code presents a well-structured way, so it was shared to other students to demonstrate they can review in code partners project easily.

VI. Threats to Validity

The studied was carried out with 53 students who are in middle of the carrier in only one university. In others academic centers the result could be different. It's necessary some replications. Currently, the tool is available only for Laravel 5 and students should know about Php and some of Laravel. The tool doesn't support extension relationship yet. Also, the tool transforms the class diagram to code, but not the other way around.

One problem was the homogeneous knowledge about software design. However, during the first 3 months, the students was formed with software design concepts, tools and MDD approach. A second problem was comprehension of the 2 cases to solve. This was mitigated with a workshop to analyze each case.

VII. Result and Discussion

After the last session using the tool, the students were asked about their experience using a custom SUS survey based on Likert where 1 is strongly disagree and 5 is strongly agree. The complete information about the study is in a OneDrive folder in https://bit.ly/2021-plantumlgen-files. A System Usability Scale (SUS) is an instrument commonly used to get the perception of usability of a product like the proposed tool in this study [37].

According to Fig. 6 of the results of the experience use survey, most users agree with the proposed tool, showing than the users like to use the tool in the future and the tool is easy to use for diagramming and keep it live during the coding. The study assumes the users had some knowledge about MVC Laravel Project and only was necessary to explain about PlantUML syntax. Also, the users know about Object Oriented Programming and got the knowledge about design phase diving during the full course.

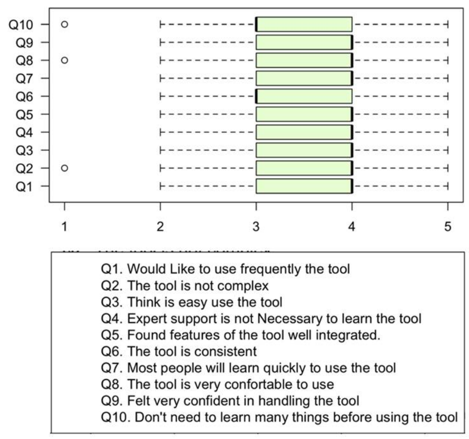


Figure 6. Box plot after experience of using the proposed tool.

However, some users had a disagree experience showing the tool needs to improve about some features collected during a feedback tool, which are documented like bugs in the tool GitHub repository. But, there are some external factors like the teaching strategies and the student interest about the topic, so the professor could find strategies to teach this kind of course using the tool.

According to Fig. 7-a about perception to use, 60% of the users indicated their willingness to use this tool frequently, while 4% disagreed. Likewise, in Fig. 7-b about perception of learning, 68% of users think that most people would learn very quickly to use the PlantUML Gen tool.

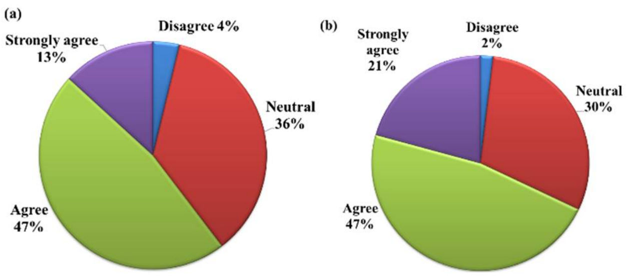


Figure 7. Perception about willingness to use (a) and Perception of learning very quickly to use Plant UML Gen tool (b).

According to Fig. 8-a about perception to ease of use, 70% of the users indicated that this tool is easy to use, while 6% disagreed. Likewise, in Fig. 8-b about perception of the tool's complexity, 62% of users think that PlantUML Gen tool is not complex.

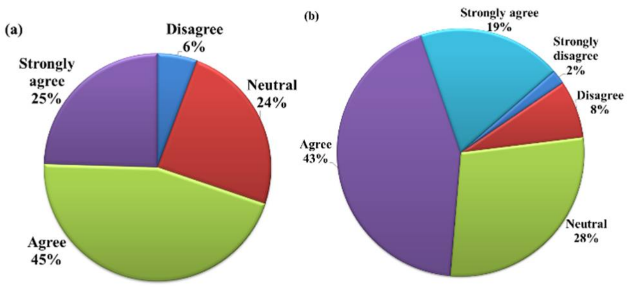


Figure 8. Perception about ease to use (a) and Perception about tool is not complex (b).

Some recent works, propose an approach and tools for having the requirements in order to understand the customer need. This stage could be manage starting with user stories for transforming to use case diagram [8] and use case scenarios [22]. From the scenarios to sequence diagram [5]. Finally, from sequence diagram to activity diagrams[6]. All these proposes using PlantUML scripts by the simple way to write diagrams. The study shows difficult for some students who doesn't understand the requirements correctly, but they should use bellow diagrams to improve the results.

For design and code process, Wutthichai [24] propose a tool for single page application based on AngularJS and generate code for front-end part. Hernandez-Mendez [23] generate code for a single page application based on RESTful methods. However this research work on a backend part of the software, the goal of this work is not about the front-end, but could use the ideas of [24] and [23] to extend the features.

Ahmad [4] proposes to generate code from some UML diagrams for some languages such as PHP, Java and .Net. However, this tool depends of Eclipse IDE and a UML profile to work. The proposed tool has the same idea, but doesn't depend of any IDE, so, in the case study, the students use Visual Studio Code, Atom and Sublime Text. Also, doesn't need a UML profile, it is based on only a detailed class UML diagram.

After the case, the tool was used in the first stages of a real project in a small company. After an analysis of the requirements, the tool generates a useful MVC code draft to start the work giving encouraging insights to use in industry, However some suggestion was given, for example, the tool should have the reverse engine feature like [21], in case any developer modify the code.

VIII. CONCLUSIONS

The model driven approach has advantages to help in the SDLC. However, for students is important to use it during software design courses. In this way keep live diagrams, especially class, E/R and sequence diagrams.

PlantUML is a popular tool used for the easy way to write diagrams in a script language that the developers are custom, thus, maintain the script into the code and open with any plain/text application. This tool is used from the requirements process until the design process. But sometimes this models are forgotten and become in dead diagrams. However with the proposed tool the class diagram can keep alive during the design, coding, testing and inclusive during the maintainability of software, thus, motivating the students to create and maintain diagrams.

The proposed tool was an easy way to apply some design concepts for students in software engineering. According to the results, the tool had a good experience using during the last part of the course. However had some things to improve. These improves must be thinking in the course and in the software industry too.

IX. FUTURE WORK

The proposal must improve to transform objects to front-end frameworks like Angular, Vue or React, using REST protocol for the back-end. It's very important considerer this kind of frameworks because help to the productivity in the industry. Is necessary to extend the proposed tool for other MVC framework like Symfony, Django, Spring Boot or MVC .Net, Also, microframeworks like Lumen or Flask. The tool should make reverse engineering like suggests some related works. Finally, the tool must be tested in case studies for get issues and improve for the software industry.

REFERENCES

[1] B. Y. P. Jalote and P. Natarajan, "The Growth and Evolution of India ' s Software Industry," *Commun. ACM*, vol. 62, no. 1, pp. pp64-69, 2019.

[2] P. Silveira *et al.*, "A Deep Dive into the Impact of COVID-19 on Software Development," *IEEE Trans. Softw. Eng.*, pp. 1–1, 2021.

[3] D. Batra, "The Impact of the COVID-19 on Organizational and Information Systems Agility," *Inf. Syst. Manag.*, vol. 37, no. 4, pp. 361–365, 2020.

[4] S. I. Ahmad, T. Rana, and A. Maqbool, "A Model-Driven Framework for the Development of MVC-Based (Web) Application," *Arab. J. Sci. Eng.*, 2021.

[5] Y. Morikawa, T. Omori, and A. Ohnishi, "Transformation method from scenario to sequence diagram," *IC3K 2018 - Proc. 10th Int. Jt. Conf. Knowl. Discov. Knowl. Eng. Knowl. Manag.*, vol. 3, no. Ic3k, pp. 136–143, 2018.

[6] M. Mythily, M. L. Valarmathi, and C. A. D. Durai, "Model transformation using logical prediction from sequence diagram: an experimental approach," *Cluster Comput.*, vol. 22, pp. 12351–12362, 2019.

[7] F. Kussunga, P. Ribeiro, and N. Santos, "Characterization of a visual environment to support SCRUM ceremonies," in *Atas da Conferencia da Associacao Portuguesa de Sistemas de Informacao*, 2019, pp. 1–10.

[8] R. S. Madanayake, G. K. A. Dias, and N. D. Kodikara, "Transforming Simplified Requirement in to a UML Use Case Diagram Using an Open Source Tool," *Int. J. Comput. Sci. Softw. Eng.*, vol. 6, no. 3, pp. 2409–4285, 2017.

[9] G. Rossi, M. Urbieta, D. Distante, and J. M. Rivero, "25 Years of Model-Driven Web Engineering. What we achieved, What is missing," *CLEI Electron. J.*, vol. 19, no. 3, 2016.

[10] D. Singh and H. J. S. Sidhu, "Optimizing the Software Metrics for UML Structural and Behavioral Diagrams Using Metrics Tool," *Asian J. Comput. Sci. Technol.*, vol. 7, no. 2, pp. 11–17, 2018.

[11] K. Ma, H. Teng, L. Du, and K. Zhang, "Exploring model-driven engineering method for teaching software engineering," *Int. J. Contin. Eng. Educ. Life-Long Learn.*, vol. 26, no. 3, pp. 294–308, 2016.

[12] S. Ouhbi and N. Pombo, "Software engineering education: Challenges and perspectives," *IEEE Glob. Eng. Educ. Conf. EDUCON*, vol. 2020-April, pp. 202–209, Apr. 2020.

[13] S. Jarzabek and H. D. Trung, "Flexible generators for software reuse and evolution (NIER track)," *Proc. - Int. Conf. Softw. Eng.*, pp. 920–923, 2011.

[14] E. de Araújo Silva, E. Valentin, J. R. H. Carvalho, and R. da Silva Barreto, "A survey of Model Driven Engineering in robotics," *J. Comput. Lang.*, vol. 62, no. December 2020, p. 101021, 2021.

[15] V. Besnard, M. Brun, F. Jouault, C. Teodorov, and P. Dhaussy, "Embedded UML model execution to bridge the gap between design and runtime," in *Software Technologies: Applications and Foundations*, vol. 11176 LNCS, 2018, pp. 519–528.

[16] J. C. Dageforde, T. Reischmann, T. A. Majchrzak, and J. Ernsting, "Generating app product lines in a model-driven cross-platform development approach," *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 2016-March, pp. 5803–5812, 2016.

[17] Y. Martínez, C. Cachero, and S. Meliá, "MDD vs. traditional software development: A practitioner's subjective perspective," *Inf. Softw. Technol.*, vol. 55, no. 2, pp. 189–200, 2013.

[18] Emmanuel Sávio Silva Freire, G. C. Oliveira, and M. E. de Sousa Gomes, "Analysis of open-source case tools for supporting software modeling process with UML," *ACM Int. Conf. Proceeding Ser.*, pp. 51–60, 2018.

[19] K. Aparna and S. Jayaraman, "Building a common notation for enabling comparison of design and execution," *2017 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2017*, vol. 2017-Janua, pp. 1609–1612, 2017.

[20] D. Kolovos, A. De La Vega, and J. Cooper, "Efficient generation of graphical model views via lazy model-to-text transformation," *Proc. - 23rd ACM/IEEE Int. Conf. Model Driven Eng. Lang. Syst. Model. 2020*, pp. 12–23, 2020.

[21] J. Yang, Y. Lee, D. Gandhi, and S. G. Valli, "Synchronized UML diagrams for object-oriented program comprehension," *ICCSE 2017 - 12th Int. Conf. Comput. Sci. Educ.*, pp. 12–17, 2017.

[22] F. Gilson and C. Irwin, "From user stories to use case scenarios towards a generative approach," *Proc. - 25th Australas. Softw. Eng. Conf. ASWEC 2018*, pp. 61–65, 2018.

[23] A. Hernandez-Mendez, N. Scholz, and F. Matthes, "A model driven approach for generating restful web services in single page applications," in *MODELSWARD 2018 - Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development*, 2018, vol. 2018-Janua, no. Modelsward, pp. 480–487.

[24] W. Chansuwath and T. Senivongse, "A model-driven development of web applications using AngularJS framework," in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science, ICIS 2016 - Proceedings*, 2016.

[25] A. Yaseen and H. Fawareh, "Visualization and Synchronization of Object-oriented Programs using Re-engineering Approach," *Int. J. Eng. Res. Technol.*, vol. 12, no. 8, pp. 1239–1246, 2019.

[26] A. Rodrigues Da Silva, "Model-driven engineering: A survey supported by the unified conceptual model," *Comput. Lang. Syst. Struct.*, vol. 43, pp. 139–155, 2015.

[27] C. Atkinson and T. Kuhne, "Model-driven development - A metamodeling Approach," *IEEE Softw.*, vol. 20, no. 5, pp. 36--41, 2003.

[28] D. Ameller, X. Burgués, O. Collell, D. Costal, X. Franch, and M. P. Papazoglou, "Development of service-oriented architectures using model-driven development: A mapping study," *Inf. Softw. Technol.*, vol. 62, no. 1, pp. 42–66, 2015.

[29] W. Cui, L. Huang, L. J. Liang, and J. Li, "The research of PHP development framework based on MVC pattern," *ICCIT 2009 - 4th Int. Conf. Comput. Sci. Converg. Inf. Technol.*, pp. 947–949, 2009.

[30] C. Liyan, "Application research of using design pattern to improve layered architecture," *Proc. - 2009 IITA Int. Conf. Control. Autom. Syst. Eng. CASE 2009*, pp. 303–306, 2009.

[31] S. Q. Huang and H. M. Zhang, "Research on improved MVC design pattern based on struts and XSL," *2008 Int. Symp. Inf. Sci. Eng. ISISE 2008*, vol. 1, pp. 451–455, 2008.

[32] S. Adam, Stenly; Andolo, "A New PHP Web Application Development Framework Based on MVC Architectural Pattern and Ajax Technology," *2019 1st Int. Conf. Cybern. Intell. Syst.*, vol. 1, no. August, pp. 18–22, 2019.

[33] H. A. Sulaiman, M. A. Othman, M. F. I. Othman, Y. A. Rahim, and N. C. Pee, "An Empirical Study of the Evolution of PHP MVC Framework," *Lect. Notes Electr. Eng.*, vol. 315, no. January, 2015.

[34] T. Ollsson, D. Toll, A. Wingkvist, and M. Ericsson, "Evolution and Evaluation of the Model-View-Controller Architecture in Games," *Proc. - 4th Int. Work. Games Softw. Eng. GAS 2015*, pp. 8–14, 2015.

[35] Y. Singh and M. Sood, "Models and transformations in MDA," *2009 1st Int. Conf. Comput. Intell. Commun. Syst. Networks, CICSYN 2009*, pp. 253–258, 2009.

[36] A. T. Rahmani, V. Rafe, S. Sedighian, and A. Abbaspour, "An MDA-based modeling and design of service oriented architecture," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3993 LNCS, pp. 578–585, 2006.

[37] R. A. Grier, A. Bangor, P. Kortum, and S. C. Peres, "The System Usability Scale: Beyond Standard Usability Testing," *http://dx.doi.org/10.1177/1541931213571042*, pp. 187–191, Sep. 2013.