A new challenge to build an application for design Generative Design Pattern

Mazen Ismaeel Ghareb
Department of Computer Science
University of Human Development
Sulimanya, Iraq
mazin.ismaeel@uhd.edu.iq
mazen.ghareb@hud.ac.uk

***Abstract*—A design pattern is used as a static reusable component of object oriented design in the many patterns catalogue. The regular design pattern does not show any collaboration of shared resource between patterns in the software design. But generative design pattern is a new design pattern that shows the relationship and shared resources between them. The generative design pattern is considered a dynamic and active design, which creating new design as a result of collaboration and resource usage between two designs. This paper will demonstrate benefit and the structure of generative pattern. It also demonstrates the creation of a desktop application for modeling generative design pattern. The Java language creates the desktop application. The application provides many features, for instance, users can place drawing objects such as class, Interface and Abstract Class object. The users also can draw different connection line between these objects, such as simple, inheritance, composition lines. This project shows the implementation details techniques of drawing objects and their connection. It also provides an open source code that many novice developers can understand and analysis for further development. The application source code gives the developers new ideas and skills in object oriented programming and graphical user interface in Java language.**

Keywords—Design Pattern, Generative Design pattern, Object Oriented Programming.

I. INTRODUCTION

The aims of this paper is to explain how to design, analysis, implement, test and evaluate a Java desktop application using new object oriented techniques which is called Generative Design Pattern. The application is capable of drawing and designing main components of generative design pattern. According to Dr. Dave he claims that this application in a few years will change the future of design in software development. It will make development process faster because it saves time and help to reuse the resources in their structure and make designing a software application more integrated.

Design object-oriented software is difficult to produce. Designing reusable object oriented software is even more difficult. This is because it needs to find the related object in the classes, and define a class, interface and inheritance hierarchies, to establish a key relationship between them. The fundamental objectives are to achieve part of this application in three stages. The application should be able to drawing UML (Unified Modelling Language) notation such Class, Interface and Abstract class. It contains the feature of delete, move and select for all drawing components. Make three types of connection between them such as simple, inheritance, and composite connections. This development of this application divided into three stages, stage one describes problem contexts, solutions and the references which are used to identify the issue. During this period, the developer needs to meet the stakeholder and gather all requirements that needed to be delivered. . Adding to that mark, an investigative research about generative design pattern and understand it is structured and the available software that capable of design UML notations. Next stage deals with designing the application. The design procedure is accomplished by using Soft System Methodology SSM [1], it illustrates the rich picture, class and sequence diagram of the application. Finally, in the third stage the implementation of the desktop application has been done by using the entire information on stage one and two. The implementation stage needs an advanced programming in object-oriented programming in Java, especially in graphics. It is recommended to conduct a research about different development method and choosing the suitable method. This project used Model Driven Architecture (MDA) development method which helps to deliver this application in three months' time [2]. The implementation stage is very challenging; it needed dealing with generic objects in the Java language to store drawing objects such as a link list. It is vital to have a good knowledge of using the Integrated Development Environment (IDE) such as eclipse or NetBeans. The application development progress deals with developing a mouse listener event in Java language because all operations of the drawing object will be accomplished by mouse events [3].

II. LITERATURE REVIEW

Design object-oriented software is difficult to produce. Designing reusable object oriented software is even more difficult. This is because it needs to find the related object in the classes, and define a class, interface and inheritance hierarchies, to establish a key relationship between them. According to [4] Fixed reused object-oriented design is hard to produce instantly. These patterns are used to solve design problem and make it more elegant, flexible and ultimate for reusing. Therefore the successful pattern will be used by designer in new design problem without the need to know about the pattern solution depending on successful usage. Each design pattern names explain, and evaluate recurring design in object oriented systems. It is easier to reuse successful design and architecture. It helps the developer to access them and choose the alternative design that improves the documentation and the maintenance of the system as well.

Christopher Alexander has important contributions in design pattern. Who had a series of books in designing urban and town plan and architecture, these books inspired programmers to design object oriented from design pattern. According to Alexander "To reach the quality without a name we must build a living pattern language as a gate "[5]. He meant by that the quality of life cannot be made but only generated. For instance, when a thing is made, it has a will of the person who make it, but when it is generated, it is freely by the operations of self-image rules acting on real situations and generated. The gang of four which they had written a design pattern book said that each pattern describes a problem that repeated over and over in our environment. Then they describe solutions to those problems. These solutions can be used millions of times over and over, but not in the same ways twice [4]. So as Christopher mentions a design pattern is a tool for improving existing code. It allows the writing of code to be easier to implement and maintain. It is used to improve efficiency and more importantly to improve the developer skills in designing and improve the qualities of products. It also allows to find a different solution to the same problem and giving a wider the scope of skills sets [6]. Generally, pattern has four vital elements problem name, problem, solution and consequences. The name is used to explain design problem, its solutions and consequences in short name one or two words. Naming the pattern increases the vocabulary of the design. It gives designer higher level of abstraction. The problem is described when the pattern applied. The context of the problem might explain specific design issue such as how to represent algorithms as objects. The problem might describe the structure of a class or object that is indicative the unbreakable design. The solution of the design describes the element that makes it, with it is responsibilities, collaborations and relationships. This is because the pattern is a template that can be applied in many different situations so the solution does not describe an implementation of the problem. The consequences are the results of applying the pattern. The key issues for consequences are evaluating design alternative and understanding the cost and benefit of using the patterns. The software development processes deal with a time trade off, space, address of implementation and language issues [4].

A. Catalogue of design pattern

A design pattern is different in their level of abstraction and difficult to understand software design. Therefore, gang of four define some of design patterns that have been used in object oriented design solutions [4]

Abstract Factory: It provides and interface for creating the depending object without specifying their main class.
Adapter: It creates another interface by converting the class interface to clients and allow classes to work together, but in reality they cannot work together because of incompatible interfaces.
Bridge: It decouples the implementation of two classes so they can vary independently.
Builder: It creates a different representation of object construction by separation complex constructed from it.
Command: It supports undoable operation by encapsulate an object. It is letting parameterize client with different requests.
Composite: It composites objects into a tree structure. Allow client to treat the object individually and compositions of objects uniformly, and represent part or whole hierarchies.
Decorator: Provide additional responsibilities' to object dynamically. It provides alternative flexible to sub classing in order to extend functionality.
Facade: It gives higher level of interface that makes the subsystem easy to use. It creates a unified interface for a set of interfaces in a subsystem.
Iterator: It helps to access elements of aggregate objects in sequence. Iterator implementation is invisible to the client.
Observer: It provides dependency between objects one-many, when one object changes its state all other dependent object informed and changes their status automatically.
Visitor: It defines new operation of element changing the classes. This operation is performed on the element of the object. There are many other patterns; therefore the best way is to organize them in order to understand how to use them. According to Gama they classify the pattern according to two criteria [4]. A first criterion is the purpose which describes what the pattern does. A second criterion is the scope which specified the pattern applies to the classes or objects. In figure 3 displays the classification of the class depending on the purpose and the scope.

		Purpose		
		Creational	Structural	Behavioural
Scope	Class	Factory Method	Adapter (class)	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter (object) Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Stratergy Visitor

Figure 3 classification of the class depending on the purpose and the scope

B. *Generative Design Pattern*

Previously it has been explained that design pattern and the relationships between these patterns. Generative Design pattern defines these relationships. For instance, pattern 'X' can use a pattern ' Y', associate with or can combine with.The result of these relationships creates new solutions in software development and create new designs from these collaborations. Dr. Wilson mentioned in his thesis that generative design pattern is a dynamic design pattern [7] The generative pattern shows how Pattern 'X' uses Pattern 'Y' and collaborates with each other. And it is better than traditional design pattern that did not show any relationship between any of two design.He also describes it as dynamic design because it allows to share resources between collaboration design patterns. Also, it will reduce time during the implementation and provide new ways of solving software development problems. For instance Composite design pattern is defined as a generative pattern when related to Decorator Patten. The problem is that the composite pattern cannot implement collection of object in different shape, size or group. The solution that the client can draw objects and composite throw interface component that is implemented by all components. The collection that stores the composite object can be an Array List or vector. The related pattern is a Decorator Pattern. It added decoration and functionality to the object. The Decoration is applied to the object rather than being a part of it. The decoration design pattern can change the decoration of the object without affecting it. This is called decoupling from the object. This decoupling has helped to divide the system into independent units. The relationships between the composite and decoration design is described. When it is implemented both would have a component interface that combine into a single interface. The composite pattern will supply the collection element from the bolt patterns, while the decorator acts as an interface for concrete decorator components [7]. Figure 4 shows the relationships between Composite and Decorator patterns.

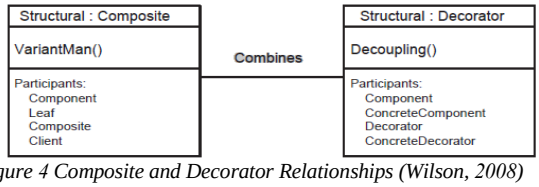

Figure 4 Composite and Decorator Relationships (Wilson, 2008)

That generative design pattern is used in generating code for parallel code for a distributed memory environment (Tan et al., 2003). It has also been used in design network applications and develops many network server applications [8].

III. METHODOLOGY

A. *Development Method for the application*

There are common points between all of development methods. The methods are meeting user needs, analysis, design and implementation of the product. All these previous methods have been used in the development, generative design pattern application. However, there are some differences of this iteration in development generative design application. This change includes adding or removing extra steps of the method in order to save time, and produces higher qualities of software. In the generative application development method used more than one development method ideas, because it's developed by a solo developer and one stakeholder. The details of the development progress of generative design pattern are as follows. The first step that is needed is to gather information about the product. It required meeting with the stakeholder Dr. Dave Wilson that he explains the requirements. What is recommended for analysis the requirements are to study his thesis about generative design pattern [8]. This application is to complex and challenging. So it has been decided to use a piece of sketcher Java application source code to save time and focus on the actual requirement of stakeholder, which is Ivor Horton's source code [10]. The whole code has been analysis and understood. Then it has been updated according to the project objectives. The development method used an idea of Rapid application development ideas to speed up the process. Choosing prioritizes some requirement of the stakeholder and completed. After stakeholder feedback who asked me to speed up the development progress alternative method that has been chosen, because RAD wastes much time in it is iteration and its complex customization. The RUP development method has not been chosen in this software iteration process, because it is too complex and costly. While the MDA model Driven Architecture have been chosen and used as development method. This is because it is suitable to apply to this application and it could be configured with new requirements. MDA first stage is to work on free open source code which considers Platform Independent Model PIM. Then it is customized to the Integrated Development Environment (IDE) such as eclipse to create new Platform Specific Model PSM. However, other ideas for developing the methods have been discussed such as prototyping development method. For example, horizontal prototyping as it is described in figure 5 which shows each task implemented horizontally and thus continue until the task is finished. Stakeholder allocates new tasks every week after evaluating and accepting the previous task. It looks like horizontal prototyping because the tasks are developed with horizontal until the end. An Idea of using

the source code as a beginning step to develop the product is taken from the extreme programming agile method. In summary, Generative Design Application uses the controversial development method, because it consists of different development method techniques. First MDA, Model Driven Architecture is used as a main development method structure. A second, horizontal prototyping technique has been used to deliver some weekly tasks. Third RAD Rapid Application Development has been used to speed up and deliver some of the requirements that requested by stakeholders. Finally, some ideas of agile method XP has been used to test many source codes and analysis them in the first stage of the application [9].

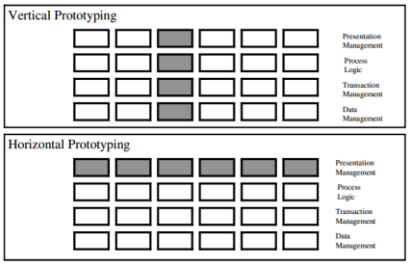

Figure 5 Vertical and Horizontal prototyping

Generative Design Pattern (GDP) is a software application that allows the user to draw many different shapes such as Interface, class, Abstract class and can save the drawing into files. Users can draw some design pattern such as Composite, Decorator and Builder Design pattern. This software is a Preliminary idea to make a dynamic design pattern. It illustrates how to link two objects as a first step in this huge software to draw generative design pattern. The aim of this project is produced much functionality such as drawing multiple shapes and illustrates the connection between them and mixed all with each other. And finally produce generative design pattern. If all functionalities of GDP are not delivered on this project, new MSc students in the future will complete it.

B. Software Development process

A soft system methodology was developed by Professor Peter Checkland in university of Lancaster is the best member of developing the SSM methodology. The Aim of SSM is to understand and analysis the problem rather than solve it. SSM concern with Human Activities System HAS. HAS deals with defining activities that people involve in and the relationships between these activities. So Checkland divided the SSM into 7 stages [11]:

1- Unstructured problem situation.
2- Expressed the problem.
3- Root definition of the system.
4- Construct a conceptual model.
5- Comparing the conceptual model with real word.
6- Describing the changes.
7- Taking action after changes is implemented.

Rich picture is used to represent stages one and two of SSM. It shows the relationships between all system components. The rich picture illustrates the problem with a simple module that all team members and stakeholder will easily understand and reflect on it. Rich Picture helps to address the problem in more details. Figure 6 shows the Generative Design Pattern Desktop Application rich picture.

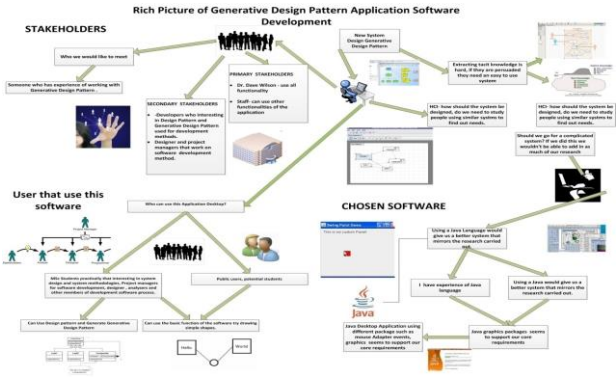

Figure 6 Rich Picture of Generative Application

Root definition is the third stage in Checkland in SSM. It defines the structure description of the system. It is a clear statement of the activities of the organization being studied. Root definition, comprises three elements (What, How, Why). For instance X-What the system does. Y-How it does it, and Z-Why is being done. To achieve these points CATWOE mnemonic helps to identify and categorize the stakeholders (People, Environment, Processes and Entities) to formulate the root definition [12].

The **CATWOE** Mnemonic is describing blow:

C-Customers or Clients: It describes the end user that receives the system and uses it.
A-Actor: It deals with (People participate in the system. Usually who are responsible for transforming input to outputs).
T-Transformation: It deals with what the system will do to the input and transform it to output.
W-Weltanschauung (World view): It deals with the world view of the system. Putting the system in wider context and highlight the consequences of the whole system.
O-Ownership: People who have the authority to decide on the future of the system. Adding to that if they want to stop the system or change the objective of the system usage.
E-Environment constrains: It deals with the environmental constraints such as financial constraint, ethical limitation, regulation, resource limitation and limit set by term of reference.

To apply these concepts on this application Design a Generative Design Pattern using Java desktop application the CATWOE is represented as follows:

C-Customers or Clients: The stakeholder is Dr. Dave Wilson, MSc Students who are interested in using and design, generative design pattern and other School staff

who their major is to deal with system development molding.

A-Actor: The actor of this system designer, developers, project managers and other members who participate in software development process and responsibilities of design the software. This software will give them different idea of traditional software design.

T-Transformation: The system will be capable of transforming design pattern to generative design pattern and show the different between them.

O-Ownership: The developers of the application that can continue to develop the software, change the objective of it, or stop this project.

E-Environment constrains in this Application there are many constraints such as financial constrain that the university could not support me with license software, resources the generative design pattern is a new subject and the resource is limited and regulated because this project must be delivered only by solo developer.

The conceptual model is the fourth stage in Checkland definition. It is derived from root definition. Conceptual Model shows the minimum activities from human activity system. All elements of conceptual model should be driven from root definition. All the elements of CATWOE must be included in the conceptual model, with strong link are grouped together. The real world can be imposed on it to reveal discrepancies. Figure 7 shows the conceptual model.

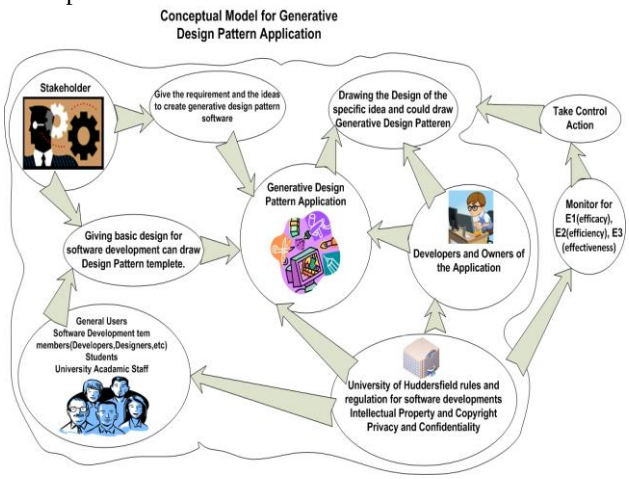

Figure 7 Conceptual Model Generative Design Pattern

As it appears in conceptual model all elements of CATWOE is existed in it. The monitor of the system is important for controlling and evaluating the application objective and purpose. Conceptual Model evaluated by these measures below:

E1 (efficacy): are the Generative Design Patterns is created?

E2 (efficiency): How many Generative Design Pattern is produced, of what standard?

E3 (effectiveness): is the qualities of the Generative Design Pattern are useful for practitioners and users? All these should be controlled and addressed to deliver good quality of this application. Stage Five and Six of Checkland will be concerned about how these activities will be carried out and compared with the real world. The Activities' might not exist in the real world. If not it requires changing the activities to meet the real world situation.

C. Results and contributions

Final Stage is deal with the recommendation for change being implemented [13] The Development process Model Driven Architecture development method has been chosen. The source code has been updated to meet the requirement which is defined as a Platform Independent Model (PIM). PIM helps to build and develop other drawing object. Later Platform Specific Model (PSM) model is generated which is a part of PIM but it contains more details about specific models. After PSM and PIM is creating a relationship source code between them is developed to deliver drawing object. The relationship between PIM and PSM is considered as the challenging stage in MDA development method. Final stage of development PSM will produce the code and then it will be tested and repeat whole process until meets the requirements as it appears in Fig. 10. However, another development method ideas are applied such as agile, Rapid Application Development and prototyping.

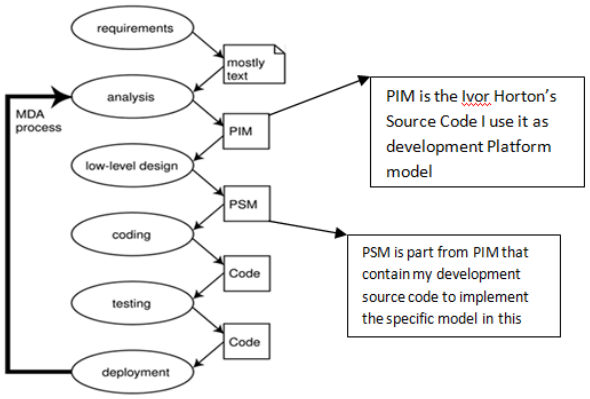

Figure 8 MDA Development method for my Generative Design Pattern Application

Finally, Graphical User Interface is used to draw Design Pattern in this application with all its functions as shown in figure 9:

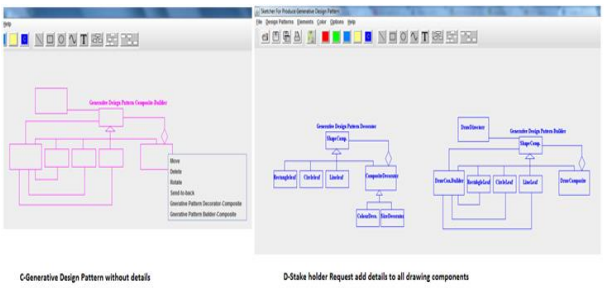

Figure 9 Creating Generative Design Pattern stages

In Figure 10, it shows the project plan for delivering this application.

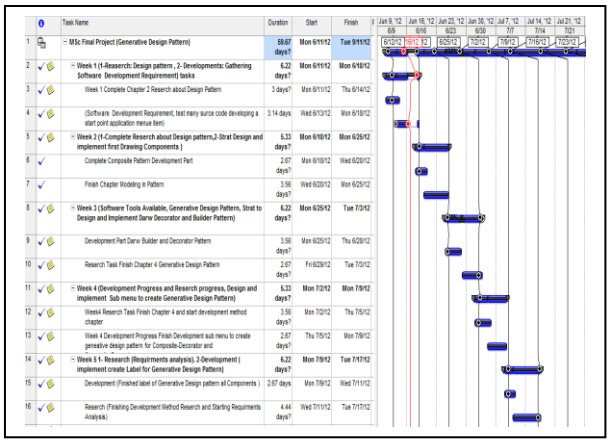

Fig. 10 Project Plan Tasks schedules details

IV. CONCLUSION

The Proposal helps me to (SWOT) analysis, which is stand for Strengths, Weaknesses, Opportunities and Threats to choose Modeling Generative Design Pattern Project. I chose this project because it is a new subject in that does not have been done before in Computing and Engineering school. It was challenging because there was not many resources in the library to achieve this project. During the three months of this project I have learned and got many skills and experience as it shows below:

- Working on advance graphical desktop application Graphical User Interface Programming (GUI). The supplication development gave me new skills of programming such as dealing link list components and working with the Abstract Windows toolkit in Java how to draw frames place many objects on it.
- Developing many mouse action listeners such as mouse pressed, mouse entered, mouse exited and mouse click. It has been used generic which is a collection of elements as array list, which contains objects of drawing components.
- I have used Integrative project model documents and lectures to manage development projects and lead the change when the implementation not meet the requirements. This model helps me in managing my project timetable.
- Swing components are used in creating the new frame that allows dealing with many different packages and powerful component to create a connection point between two objects. It helps me to learn how to deal with hash map and how to call a method of executing commands using it. It is obvious from the project that it is challenging because of dealing with dynamic graphics objects to implement a new way of drawing and designing generative design pattern.
- I have experienced in using Unified Model Language UML Diagram to develop it by installing new plug-in to eclipse tool called Object Aid. It allows converting the source code to UML notation which is saving time and accomplish the tasks quicker [14]
- I have used much development research methodology which gave me good experience to apply many development methods in future implementation for my carrier.
- The research that I have done with stakeholder gave me the abilities to challenge any other subject in the future.

The Stakeholder accepts the proposal and like it, because it displays enthusiasm and challenging to deliver Generative Design Application. He is satisfied with the project and the applications. Modelling Generative Design pattern application approved by stakeholder. He has a positive feedback as a starting point application that delivered in this short period.

Acknowledgment

I'm thanking the University of Human Development Staff also all postgraduate students of Huddersfield University. Thanks to Dr. Dave Wilson for his support.

References

[1] B. Bergvall-Kåreborn, A. Mirijamdotter, and A. Basden, "Basic principles of SSM modeling: an examination of CATWOE from a soft perspective," *Systemic Practice and Action Research*, vol. 17, no. 2, pp. 55–73,2004.

[2] A. G. Kleppe, J. Warmer, W. Bast, and M. Explained, "The model driven architecture: practice and promise," 2003.

[3] P. Keegan, L. Champenois, G. Crawley, C. Hunt, and C. Webster, *NetBeans (TM) IDE Field Guide: Developing Desktop, Web, Enterprise, and Mobile Applications*. Prentice Hall PTR, 2006.

[4] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[5] C. Alexander, *The timeless way of building*, vol. 1. Oxford University Press, 1979.

[6] C. G. Lasater, *Design patterns*. Jones \& Bartlett Publishers, 2010.

[7] D. Wilson, "A Framework for the Definiton of a Generative Design Pattern," 2008.

[8] Z. Guo, J. Schaeffer, D. Szafron, and P. Earl, "Using generative design patterns to develop network server applications," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, 2005, p. 178a–178a.

[9] P. Beynon-Davies, D. Tudhope, and H. Mackay, "Information systems prototyping in practice," *Journal of Information Technology*, vol. 14, no. 1, pp. 107–120, 1999.

[10] A. Platt and S. Warwick, "Review of soft systems methodology," *Industrial Management \& Data Systems*, vol. 95, no. 4, pp. 19–21, 1995.

[11] E. del Nuevo, M. Piattini, and F. J. Pino, "Scrum-based methodology for distributed software development," in *Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on*, 2011, pp. 66–74.

[12] A. W. Brown, M. Delbaere, P. Eeles, S. Johnston, and R. Weaver, "Realizing service-oriented solutions with the IBM rational software development platform," *IBM systems journal*, vol. 44, no. 4, pp. 727–752, 2005.

[13] M. Völter, T. Stahl, J. Bettin, A. Haase, and S. Helsen, *Model-driven software development: technology, engineering, management*. John Wiley \& Sons, 2013.

[14] L. Forite and C. Hug, "FASMM: Fast and Accessible Software Migration Method," in *Research Challenges in Information Science (RCIS), 2014 IEEE Eighth International Conference on*, 2014, pp. 1–12.