A Model Driven Approach for Generating Angular 7 Applications

https://doi.org/10.3991/ijes.v8i2.14131

Mouad El Omari (✉), Mohammed Erramdani, Abdelkader Rhouati
University Mohameed First, Oujda, Morocco
elomari.mouad@gmail.com

Abstract—The proliferation of language and framework updates and the appearance of new ones has made the need for code generation tools an inescapable one. For this reason, many companies have started to invest in this area with the aim of perpetuating the sssknow-how.MDA-Model Driven Architecture- has enabled semi-automatic generation of code via models. The MOFM2T standard is independent of the generated language, but to date, no generator of Angular code from a UML diagram has never seen the light of day, the objective of this article is to propose a platform allowing from a class diagram to generate an operational application in Angular 7.

Keywords—MOFM2T, Angular;MDA; models; frameworks;UML diagram,generation of code.

1 Introduction

It has been almost 3 decades that researchers in the IT field began to put the conceptual bases of the platforms which will allow to pass either models to new models or models to code, the OMG (Object Management Group) is a part of it by introducing MDA as a remedys[1].

From its creation in the 1990s to the present day, the Web has evolved enormously.

Originally designed to share static documents on the Internet, today we can use real applications directly from our browser. Connectionless use, push notifications, interactions with the device (access to orientation, access to the camera), etc., most of the functionalities historically available on thick client applications are now available on the Web.

Therefore, the browsers JavaScript engines and the HTML APIs had to evolve. Browsers now offer highly optimized engines to execute JavaScript code quickly even on low-end devices. The HTML APIs have become more complete, especially with the arrival of HTML 5, which has brought a lot (storage in the browser, CSS 3 animations, etc.). This allows offering many more features.

The tools have also changed. JavaScript has become a predominant language, whereas before it was used only sparingly, many frameworks and IDE were born in order to offer a better development experience. Node.js, created in 2009, allows executing JavaScript code outside a browser and has quickly become an essential tool in

front-end development. Frameworks like React, Angular, Vue.js and many others have been created to facilitate the creation of web applications.

From 2005, the AJAX system (Asynchronous JavaScript and XML) allows interactions between the user and HTTP backend: it is finally possible to exchange the information and generate content from these interactions.

In 2010, the first version of AngularJS was launched. It makes it easier to create Single Page Applications, web applications that mimic native applications: no browser refresh, reduced loading times, a much less "internet" UI, etc. This version already allows you to do many things, but suffers from a rather complex syntax as well as the limitations of JavaScript. This is why Google chooses to completely rewrite the framework for its version 2. Today, we are at Angular8 (now simply called "Angular"); version 3 having been skipped for semantic reasons quite simply.

Our methodology to reach this objective will be articulated on the use of MDA by having a class diagram in input, this one will undergo a transformation guaranteeing the generation of an application in Angular.

This manuscript is started by the exhibition of works in this area, after we moved to a dissection of the concepts of MDA, before introducing our concept of code generation and after we ended with the conclusion and perspectives.

2 Related Work

Due to the proven benefits of MDA, many methods have emerged in the field of web engineering. Several methods have also been implemented to support their ideas such as:

AndroMDA: AndroMDA takes as input a commercial model specified in UML ("Unified Modeling Language") and generates important parts of the layers necessary for building a Java application. AndroMDA's ability to automatically translate high-level business specifications into quality code saves considerable time in the implementation of Java applications [2].

WebML: WebML provides designers with a platform that allows them to express the main functionality of a site at a high level, without engaging in detailed architectural details[3].

OOHDM: The object-oriented hypermedia design method is a model-based approach to creating hypermedia applications. It includes four different activities, namely conceptual design, navigation design, abstract interface design and implementation[4].

UWE: The UWE approach provides domain-specific notation, a model-driven development process, and tool support for web application engineering [5].

ArcStyler: Enables companies to create and integrate software applications in a highly automated and industrialized environment: it bridges the gap between business and technology by automatically transforming business models into functional software applications [6].

We tried by the same concept to generate an application in AngularJS through UML profiles [7]

3 Model Driven Engineering

3.1 The OMG approach

The concept of model in IDM explicitly refers to the definition of the languages used to build them. The language in which this model is expressed must therefore be explicitly defined. This modeling language is called meta-model.

A meta-model is a model that defines the expression language of a model, that is, the modeling language. The notion of meta-model leads to the identification of a relationship, linking the model and the language used to build it, called "it conforms to". A model is said to conform to a meta-model if each of its elements (objects and relations) corresponds to an element of the meta-model, and if it respects all the properties (for example, invariant constraints) expressed on the meta-model.

It is on these basic principles that the OMG (Object Management Group) relies to define all of its standards, in particular UML[8].

MDE (Engineering Directed by Models) is on the way to become the new paradigm in terms of application development. The objective behind the use of such a paradigm is to increase productivity and reduce time development of complex systems by means of models which are much less related to technology and which are much closer to the industry. This abstraction of complex problems makes systems easier to specify and maintain [9][10][11].

One of the best-known variations of MDE is the MDA (Model Driven Architecture) standard which, through the use of three levels of abstraction, aims to separate the business logic of the application of the technology that will be used to achieve it. The objective is to remove the direct link between applications and their coding associated, facilitating their interoperability and thus making them less sensitive to changes technologically [12].

In the field of MDE, meta-modeling plays a very important role. Indeed, it is considered a common technique of defining the abstract syntax of models and interrelationships between the elements of the model. If the model is an abstraction of the elements of the real world, the meta-model represents yet another abstraction, defining the properties of the model itself [13].

This approach emphasizes the development of models of higher level of abstraction and promotes the approach of transformation from one model to another. MDA recommends the development of the three types of following models:

- **Computation Independent Model (CIM):** This model represents the highest level and describes the system requirements and how it works in its environment while details of the application structure and achievement are hidden or even undetermined
- **Platform Independent Model (PIM):** This model describes the details of the system without showing specific ones to an execution platform or a particular technology
- **Platform Specific Model (PSM):** This model describes the details and characteristics deleted from the PIM. It must be adapted to specify the implementation of the system in a single technological platform

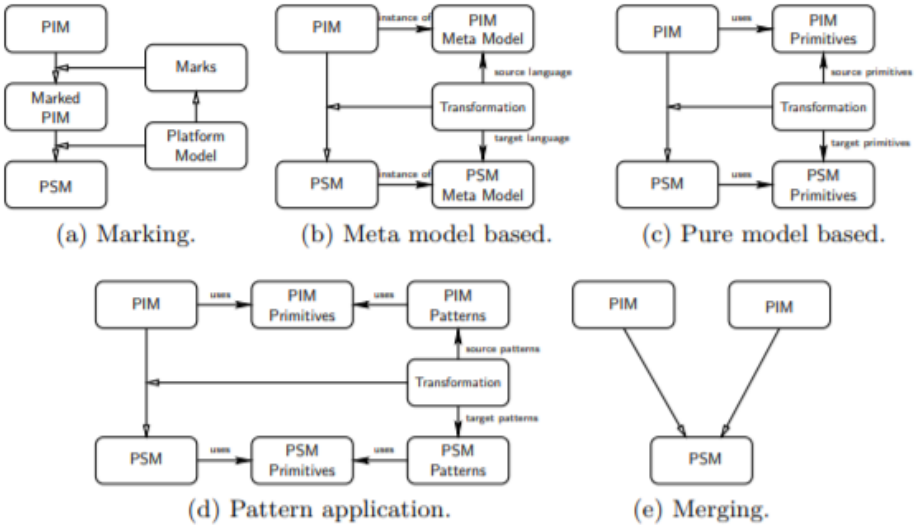

Fig. 1. Transformation strategy

3.2 Transformation of MDA models

Model transformations are at the heart of the model-driven engineering approach. But there is not yet a consensus on defining and implementing a transformation. In the literature, multiple approaches are proposed.

To carry out model transformations, these must be expressed in a certain modeling language or meta-model. Starting from the source and target meta-models of the transformation, there are two types of transformations:

Endogenous and exogenous transformations. A transformation is said to be endogenous if the models involved come from the same meta-model. But when the source and target models are of different meta-models, the transformation is said to be exogenous or else translation [14].

Models and meta-models often have a graphical representation related to a graph. A model, in this case, can be considered as a labeled graph, constrained by consistency rules essentially defined as a MOF meta-meta-model. Graph-rewriting systems combine graphical notation and textual notation to express these transformations. A transformation program essentially composed of rewriting rules will first select a fragment of a source graph identified using a navigation language [15].

4 AngularGen Workflow

I will expose our code generator that we have designed and developed, it will cover the transformation developed between the PSM and the code, AngularGen is our code generator implemented to cover this need. The input of this generator will be a class

diagram, which will be translated into XML, the latter will be parsed and thanks to the design pattern interpreter to generate an application in Angular versions 7.

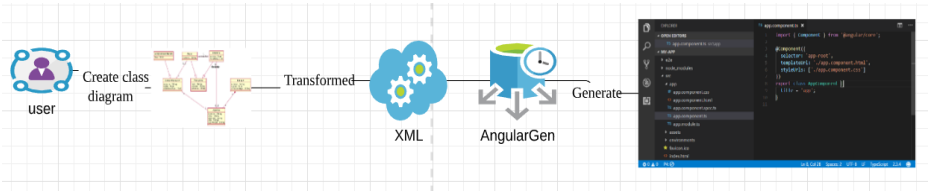

Fig. 2. AngularGen Workflow

The first phase was to define a source meta-model, of the class diagram as shown in Figure 3.

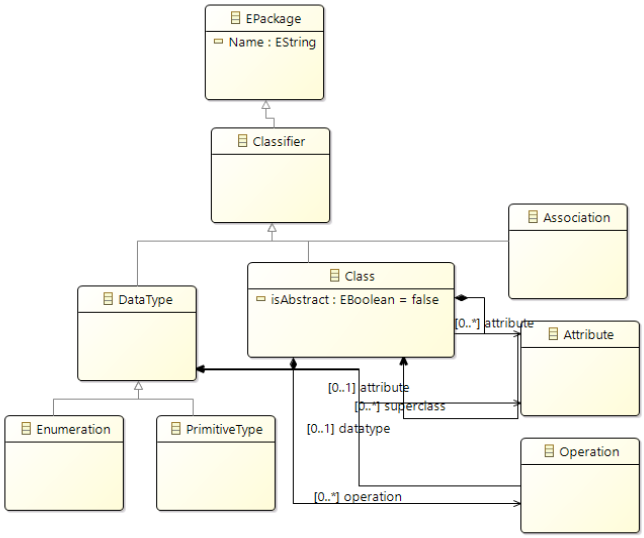

Fig. 3. Our proposed source meta-model

Package: Groups all classes, Datatype and Association.
Classifier: Collect all common elements that specialize in aClass or Datatype.
Class identified by name and Boolean to check if it is Abstract and contains attributes, associations and operations.
Attribute: Contains data that belongs to a class.
Operation: Contains the business logic of a class.
Association: Describe relationship between classes.
Enumeration: Collect datatype of enumeration.
Primitive Type: Collect all primitive types such us Integer, String, Boolean
The table above summarizes the transformations carried out on the various elements of the class diagram and their correspondents in the generated application.

Table 1. Correspondence table between AngularGentopic and class diagramtopics

Class topic	AngularGen topic
Class	Module
Class	Component
Class	Directive
Class	Interface
Class	Service
Class	Template
Class	Style file(scss)
Class	Route
Attribute	Interface attribute
Attribute Type	Interface dataType
Attribute	Template element Type
Association	dependency injection in constructor
Generalization	child component
Enumeration	Enumeration
Abstract Class	Abstract Interface
Operations	Component operation

For example, for any class of the source model, a module, component, route, service, template, style file, directive will be generated with the same name.

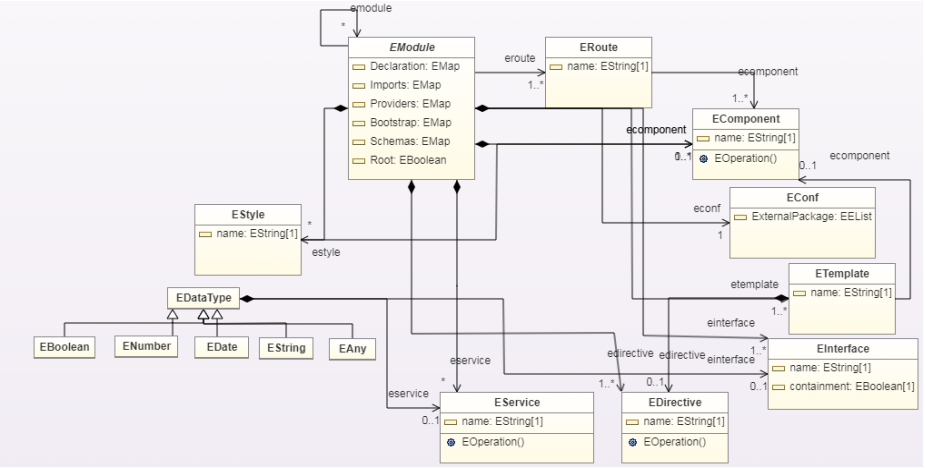

Fig. 4. Our proposed target meta-model

Each application has a root module, the latter defines the descending modules, when the ngDoBootsrap is triggered all these modules will be loaded, each module imports its services, providers, diagrams and components [16].

A module can be considered as container for different parts of application like components, directives, services.

Angular applications must be developed as a hierarchy of Components. Each Component is an isolated part of the application for maintenance reasons.

Initially, Angular is a framework for developing One-Page applications [17]. Therefore by definition, there is no need to change pages (since there is only one) and the routing is not useful [18]. However, as the growing framework and its uses became more diverse, it's soon become possible to integrate a routing system. In versions prior to 1.2, which were beta versions, developers natively integrated this routing system to the framework for the sake of simplicity (they had many other priorities). However, as of version 1.2 which is stable and which made the renown of AngularJS, it was necessary to refocus it on these primary objectives: these famous applications One-Page. As the project progressed, many functionalities were born and the team decided to separate those weren't essential, a certain page load in AJAX and the result will be added in a portion of the DOM. Template collects a set of directives which will be added to the DOM when loading the component [19].

For our code generator, we will define static rules for the configuration files, suddenly all the generated applications will have the same configuration files, the version of the packages @ angular / * installed is "~ 7.2.14".

Table 2. Workspace configurationfile

Static file or folder	Description
angular.json	The default CLI configuration settings for all projects in the workspace, including configuration options for generation, service, and testing tools that the CLI uses, such as TSLint, Karma, and Protractor
package.json	Configures the dependencies of the npm package which are available for all projects in the workspace
package-lock.json	Provides version information for all packages installed in node \ _modules by the npm client
node_modules	Contains all dependencies defined in package.json
tsconfig.json	Contains typescript config
tslint.json	Contains coding rules in typescript

5 Study Case

As I mentioned in the previous sections, the class diagram will be the entry point of our code generator, the majority of class diagram editors guarantee the conversion of the class diagram into XML.

We will take this diagram as an example:

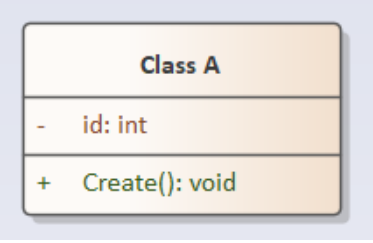

Fig. 5. Class Diagram exemple

For this class diagram, the xml file obtained:

```
<UML:Classifier.feature>
   <UML:Attribute name="id" changeable="none"
      visibility="private" ownerScope="instance"
      targetScope="instance">
   </UML:Attribute>
   <UML:Operation name="Create" visibility="public"
       ownerScope="instance" isQuery="false"
      concurrency="sequential">
      <UML:ModelElement.taggedValue>
         <UML:TaggedValue tag="type" value="void"/>
      </UML:ModelElement.taggedValue>
   </UML:Operation>
</UML:Classifier.feature>
```

Fig. 6. Generated XML

By loading this xml file into our AngularGen platform, the transformation rules cited in table 1 will be established, and an application in Angular 7 will be generated.

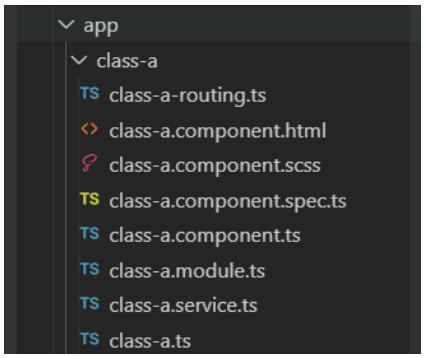

Fig. 7. Generated application architecture

As seen in the figure, a new file is created containing the name of the class and listing all the modules, services, component, route, template, style, interface files.

6 Conclusion

In this paper, UML class diagram concepts have been taken as the basis on which we can ensure the generation of an application, the use of an UML class diagram was the way with which we finally proceeded to generate our application, and such a methodology will ensure a gain on the time and durability side.

As a perspective, we want to extend what our code generator produces, by adding pattern publish subscribe between linked classes and adding the lazy loading mechanism between entities that have inheritance relationships, in addition to an automatic generation of elements of the graphical interface

7 References

[1] Fain, Y., &Moiseev, A. (2016). Angular 2 DevelopmentwithTypeScript. Manning Publications Co.

[2] AndroMDA Core Team. Andromda, 2006..

[3] Stefano Ceri, PieroFraternali, and Aldo Bongio. Web modeling language (webml) a modeling language for designing web sites.Computer Networks, 33(1-6) :137–157,2000. https://doi.org/10.1016/s1389-1286(00)00040-2

[4] SCHMID, Hans Albrecht et DONNERHAK, Oliver. OOHDMDA–an MDA approach for OOHDM. In : International Conference on Web Engineering. Springer, Berlin, Heidelberg, 2005. p. 569-574. https://doi.org/10.1007/11531371_71.

[5] KRAUS, Andreas, KNAPP, Alexander, et KOCH, Nora. Model-Driven Generation of Web Applications in UWE. MDWE, 2007, vol. 261.

[6] OBJECTS, I. ArcStyler MDA tool (http://www. arcstyler. com/). Retrieved, 2006, vol. 8, p. 2006.

[7] OMARI, Mouad EL, ERRAMDANI, Mohammed, et RHOUATI, Abdelkader. Getting Model of MVVM Pattern from UML Profile. International Journal of Recent Contributions from Engineering, Science & IT (iJES), 2020, vol. 8, no 1, p. 36-47. https://doi.org/10.3991/ijes.v8i1.13037

[8] Bézivin, J., &Gerbé, O. (2001, November). Towards a precise definition of the OMG/MDA framework. In Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001) (pp. 273-280). IEEE. https://doi.org/10.1109/ase.2001.989813

[9] Gervais, M. P. (2002, August). Towards an MDA-oriented methodology. In Proceedings 26th Annual International Computer Software and Applications (pp. 265-270). IEEE. https://doi.org/10.1109/cmpsac.2002.1044561

[10] El Omari, Mouad &Erramdani, Mohammed &SaidaFilali(2016)Model to Model Transformation by Modeling Getting Model of MVVM pattern from UML Models .Proceedings of the International Workshop on COmputing Sciences (WCOS'16), December, 21-22, 2016, Kenitra, Morocco. https://doi.org/10.3991/ijes.v8i1.13037

[11] Mellor, S. J., Scott, K., Uhl, A., & Weise, D. (2004). MDA distilled: principles of model-driven architecture. Addison-Wesley Professional.

[12] Bézivin, J., Brunette, C., Chevrel, R., Jouault, F., &Kurtev, I. (2005, October). Bridging the generic modeling environment (GME) and the eclipse modeling framework (EMF). In Proceedings of the Best Practices for Model Driven Software Development at OOPSLA (Vol. 5).

[13] Meliá, S., Kraus, A., & Koch, N. (2005, July). MDA transformations applied to web application development. In International Conference on Web Engineering (pp. 465-471). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11531371_59

[14] EL OMARI, Mouad, ERRAMDANI, Mohammed, et FILALI, Saida. Getting Model of MVVM pattern from UML Models.

[15] Ross, J.. (2004). Review: Model Driven Architecture. The Computer Bulletin. 46. 31-31. 10.1093/combul/46.1.31. https://doi.org/10.1093/combul/46.1.31

[16] El Omari, Mouad &Erramdani, Mohammed &Hajbi, Rachid. (2017). For Formed Entrepreneurial Culture. 10.1007/978-3-319-46568-5_47.

[17] FAVRE, Liliana. Formalizing MDA-based reverse engineering processes. In : 2008 sixth international conference on software engineering research, management and applications. IEEE, 2008. p. 153-160. https://doi.org/10.1109/sera.2008.21

[18] Kardoš, M., &Drozdová, M. (2010). Analyticalmethod of CIM to PIM transformation in Model Driven Architecture (MDA). Journal of information and organizational sciences, 34(1), 89-99.

[19] Moreno, N., Romero, J. R., &Vallecillo, A. (2008). An overview of model-driven web engineering and the MDA. In Web Engineering: Modelling and Implementing Web Applications (pp. 353-382). Springer, London. https://doi.org/10.1007/978-1-84628-923-1_12

8 Authors

Mouad El Omari is pursuing his PhD at Mohamed first University at MATSI laboratory. He graduates from ENSA (High School of Applied Science) as a computer science engineer. His research activities at the MATSI Laboratory (Applied Mathematics, Signal Processing and Computer Science) are focused on MDA (Model Driven Architecture) approach applied to dynamic generation of code.

Mohammed Erramdani teaches the concept of Information System at Mohammed First University. He got his thesis of national doctorate in 2001. His activities of research in the MATSI Laboratory (Applied Mathematics, Signal Processing and Computer Science) focusing on MDA (Model Driven Architecture) integrating new technologies XML, EJB,MVC, Web Services, etc.

Rhouati Abdelkader is Phd researcher in a private laboratory of Novelis Company in Paris. He got his thesis of national doctorate in 2019 from Mohamed first University Morocco. His main subject of research was about Machine learning and its application in software engineering.

Article submitted 2020-03-08. Resubmitted 2020-04-15. Final acceptance 2020-04-15. Final version published as submitted by the authors.