

JUDO



Modeling in software engineering
and its advantages for developers



Table of Contents

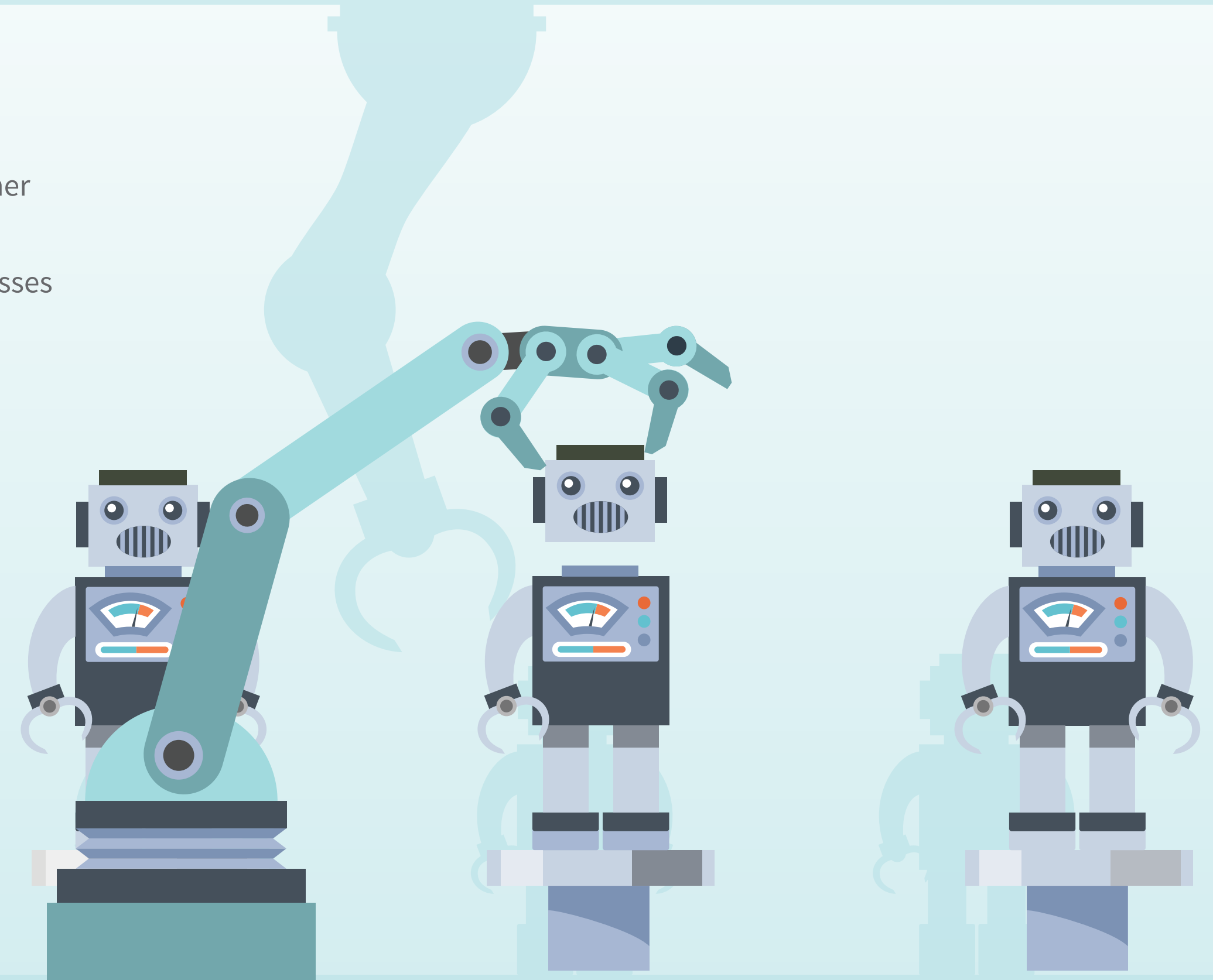
Growing Challenges	3
Introducing modeling and model-driven development	4
Domain Specific Language (DSL)	5
Meta-modeling and meta-metamodeling	6
Model-driven architecture	7
The realm of low-coding	8
The process of modeling	9
Model transformation	10
How BlackBelt creates domain models?	10
JUDO - BlackBelt's project development philosophy	11
Why modeling is useful for developers and business individuals?	12
The advantages of modeling and MDA over traditional manual coding	12
Welcome to BlackBelt, a professional application development company	13

Growing challenges

In today's world, software development is a key factor in the life of enterprises. It doesn't matter if we're talking about the implementation of in-house workflows, the sales of SaaS (Software as a Service) or PaaS (Platforms as a Service) or any other online solution, software engineering occupies a primary role within the company. Our world is approaching the trend of Software Defined Everything (SDx), meaning that the software-driven operation of certain hardware or other manual processes are becoming more and more essential and companies are being forced to face the demanding challenge of automation. Prepared and innovative competitors who quickly adopter high-tech solutions can seriously impact the customer base of other companies in the same niche. Those enterprises that don't follow the trends of software engineering can easily lag behind the competition and lose a significant part from their market share.

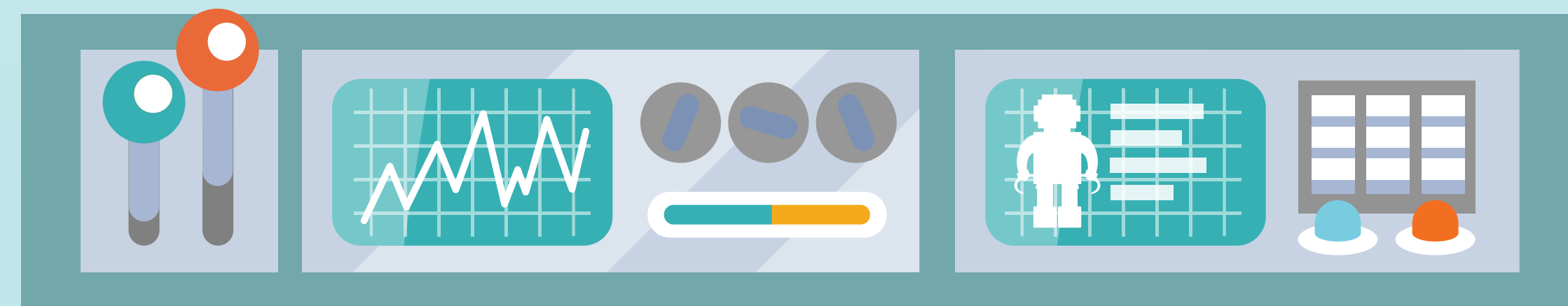
Unfortunately, traditional manual coding consumes too much time from developers' working hours and the risk of unwanted human error remains quite high. This results in slow project development, lagging communication within the company and unmet deadlines. There is an increasing requirement for an innovative solution, one which is able to speed up the pace of the whole software engineering process without letting the outcome lose quality.

To continue down this line even further, often business analysts or professionals, who have a say in the technological outcome of the development, lack the proper coding knowledge to transparently and holistically explain their expectations and requirements to the developers.



As a result, even the best programmers sometimes face situations when they misunderstand requests and the finished product does not fully align with the original design.

These are two major problems in software development. However, this is the point where modeling, model-driven architectures, and low-coding come into the picture and save the day!





Introducing modeling and model-driven development

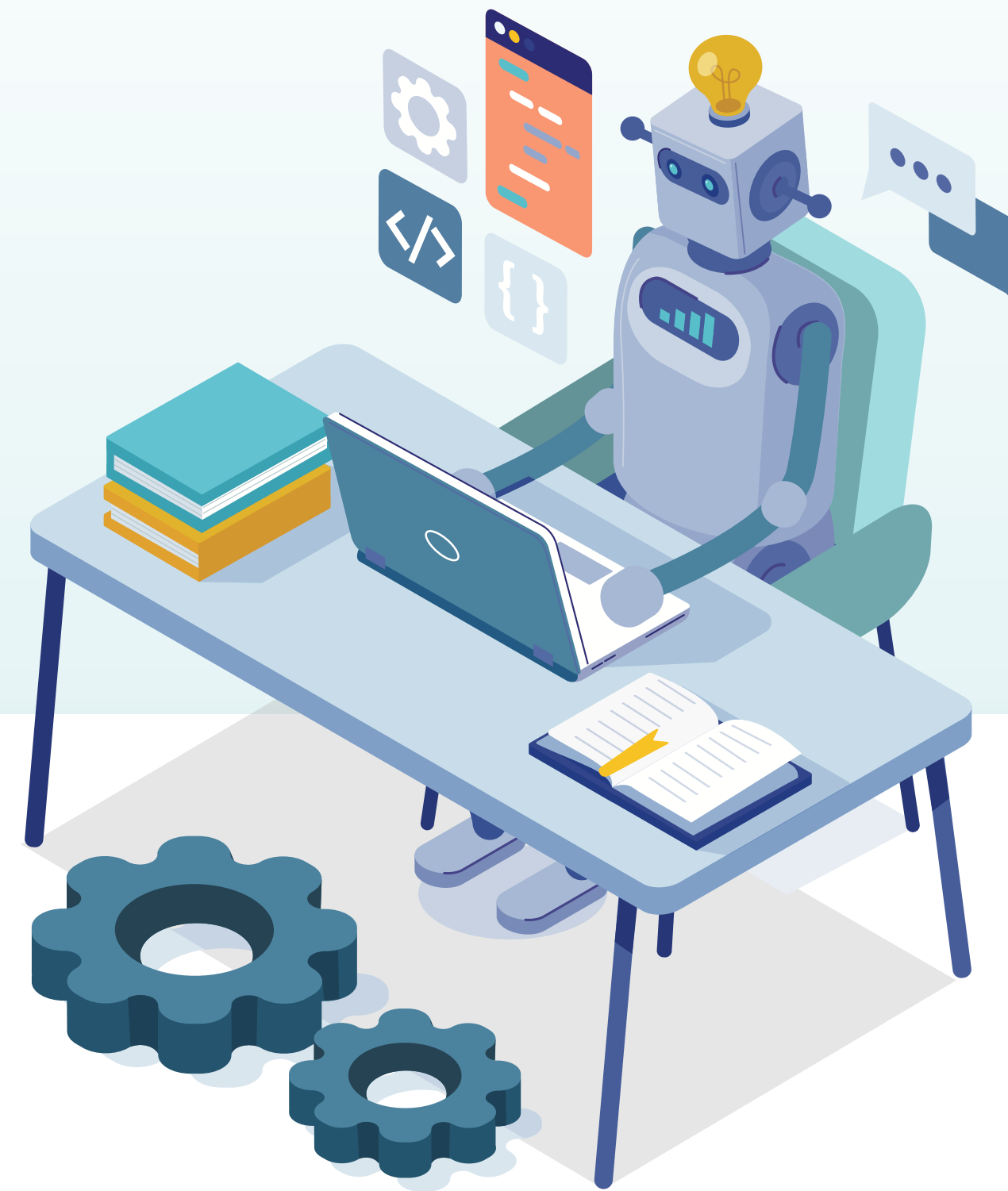
Let's first lay the foundations of this topic before we dive deeper into it. A model is a representation of a system that will be developed. Its purpose is to assist developers and professionals from the business domain to visualize the new system in a simplified way. The model is also used to reprioritise certain aspects of the complete system. As a result, those involved can consider the most significant and relevant questions about the system's logic.

In other words, a model is an external and explicit representation of reality, one that the people aim to redesign, transform or explore. The aim of modeling is to put businesses and developers on the same page and to answer all questions that may occur during the meetings and negotiations.

Even though developers are highly skilled in programming languages, they may not fully understand the objectives of the business domain during discussions. Consequently, they will need a common modeling language to facilitate fluent and smooth communication between the two parties for the successful implementation of the project and the viability of the deadline.

Domain Specific Languages (DSL)

Domain Specific Languages is a subset of languages that can be used between the business domain and the developers to communicate with each other about the specifications of the project. It is basically a computer language, that is specialized for a certain domain, like HTML for websites. Using DSLs instead of the generally used language of the developers is advantageous in modeling, because this way domain experts can understand, validate, or modify the system. Due to their limited scope, they are easier to learn. As a result of utilizing DSL in modeling, the amalgamation of business and IT knowledge will finally lead to a complete IT system.



These modeling languages are only able to describe a narrow world, however it is in a lot more detailed and punctual manner. If those involved can successfully work together to determine the details, idioms and requirements of the system, then the developers are able to write source codes with their utilization.

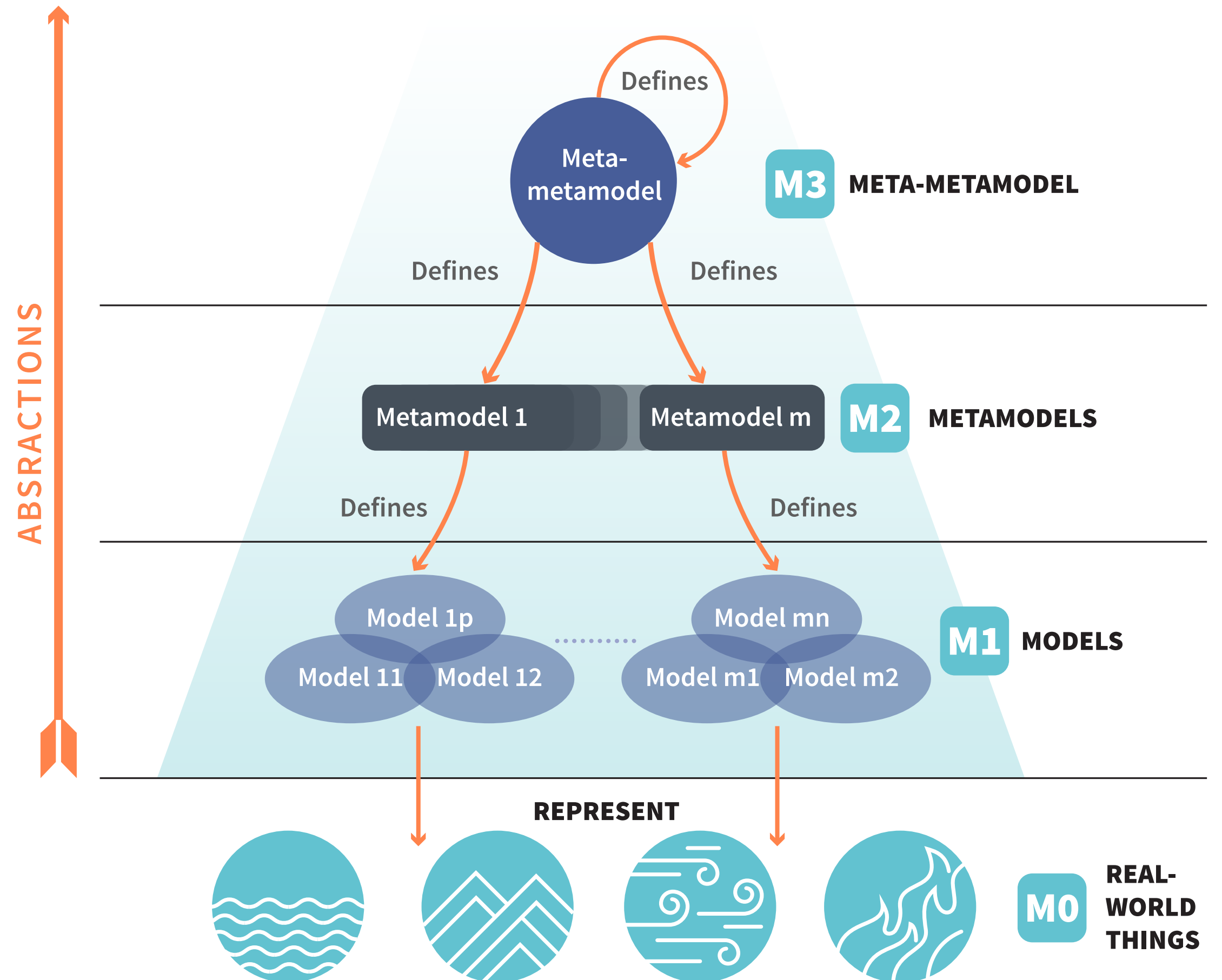
Meta-modeling and meta-metamodeling

After covering the topic of models and DSL, we need to talk about meta-modeling and meta-metamodeling. The word “meta” means beyond or above, therefore a meta-model is a model of a model. However, it is important to note that a meta-model is not a simplified or aggregated view of the previous model. It is a model, that uses a different level of abstraction to examine and make statements about the model.

The language that is necessary for the definition of the meta-model is called meta-metamodel. It is significant to mention that the distinction of meta-model and meta-metamodel is a man-made concept, one which was invented to better understand the language and its specifications. For instance, there are programming languages, like Lisp, where there is no distinction between code and data, so modeling is not possible.

The usual process of model construction first requires building the meta-model which can only then be used to build instance models.

To create successful DSL meta-models, such an environment is necessary that can be written in a DSL language, which again needs a level of abstraction or language syntax that we can use to construct the DSL.



Let's look at an example

You are at a classical music concert, where the pianist is playing a beautiful song on the piano. To write the song down the artist uses musical notation, the language of the notes. Apart from this, he/she will need

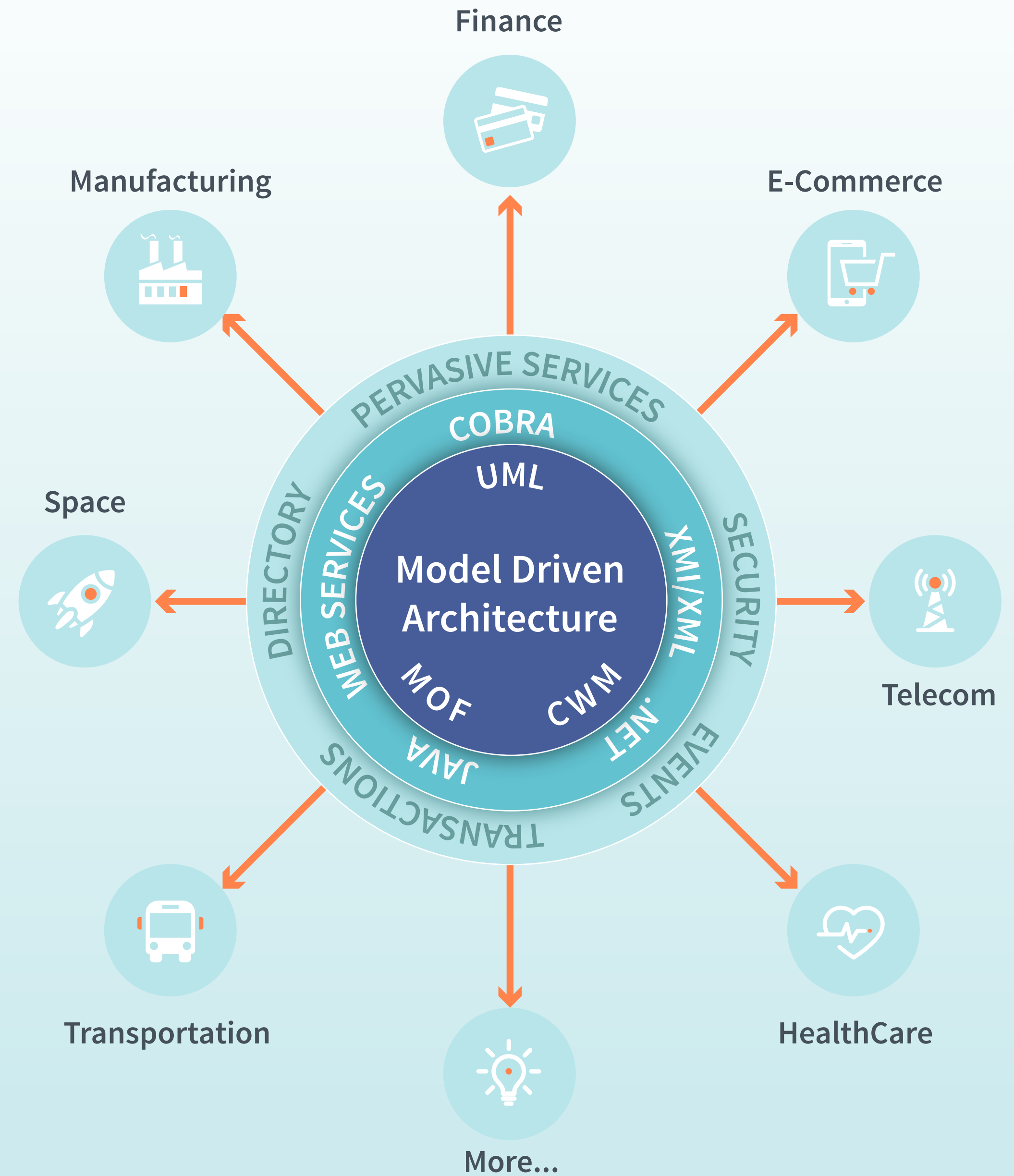
an environment, in which these notes can be properly understood and described. This is the system of the musical notes. When these are all done, the beautiful piano song can be passed on to other artists to play.

Model-driven architecture

Let's examine this a little bit further. Model-driven architecture (MDA) is a system that is created using generalized models written in the modeling language. Its objective is to provide guidelines for structuring software specifications. It effectively separates the specification of a designed system's functionality from the underlying platform technology, as a result, developers and business professionals can observe its progress, any possible inconsistencies, errors or warning signs together.

With the use of a model-driven architecture, people working on the same project can envisage the platform-independent system in order to realize the functional requirements and to see whether the logic will survive the execution process.

By having distance from platform-specific requirements, those in the business domain and those in the IT Department can successfully work together to achieve the ultimate goal of the project without unfilled gaps or errors, using a method that is understandable for both parties.



The realm of low-coding



Even though we have looked at a software engineering solution that facilitates the cooperation between the business domain and developers and found a solution to one of the most significant programming problems, there is still a need to address the question of how to increase the quantity of the output without quality loss.

Low-coding relieves companies from the burden of constant digital transformations. Low-coding is a solution where business professionals and programmers can work together to compile a fully functional web or mobile-friendly web application using pre-made elements. Usually, low-coding platforms are visual solutions that enable the individuals working on the project to drag-and-drop the chosen element from a library into the model without having to stress about codes.

It is a development platform on an abstract level that uses a DSL to deliver a meta-model to the end-user. Therefore, it limits its usability, for instance for the planning of databases, business workflows, user interface, or other web applications. By using this approach, development time can be seriously cut, the amount of output can be increased. Moreover, the possibility of human error is significantly decreased.

The utilization of low-coding is beneficial for companies because even business managers can take a look into the workflow and suggest modifications, even if they do not have any programming knowledge. At the same time, a solid SQL knowledge is enough from the developers who are able to deploy the built application into the company's infrastructure. Low-coding apps are highly customizable, they enable endless branding, integration, and refactoring.

With the help of this technology, enterprises are able to distinguish themselves from the rest of the pack, get ahead of the customer expectation curve, introduce brand new and disruptive products, and last but not least, rapidly engage and respond to customer or in-house requests across any platform.

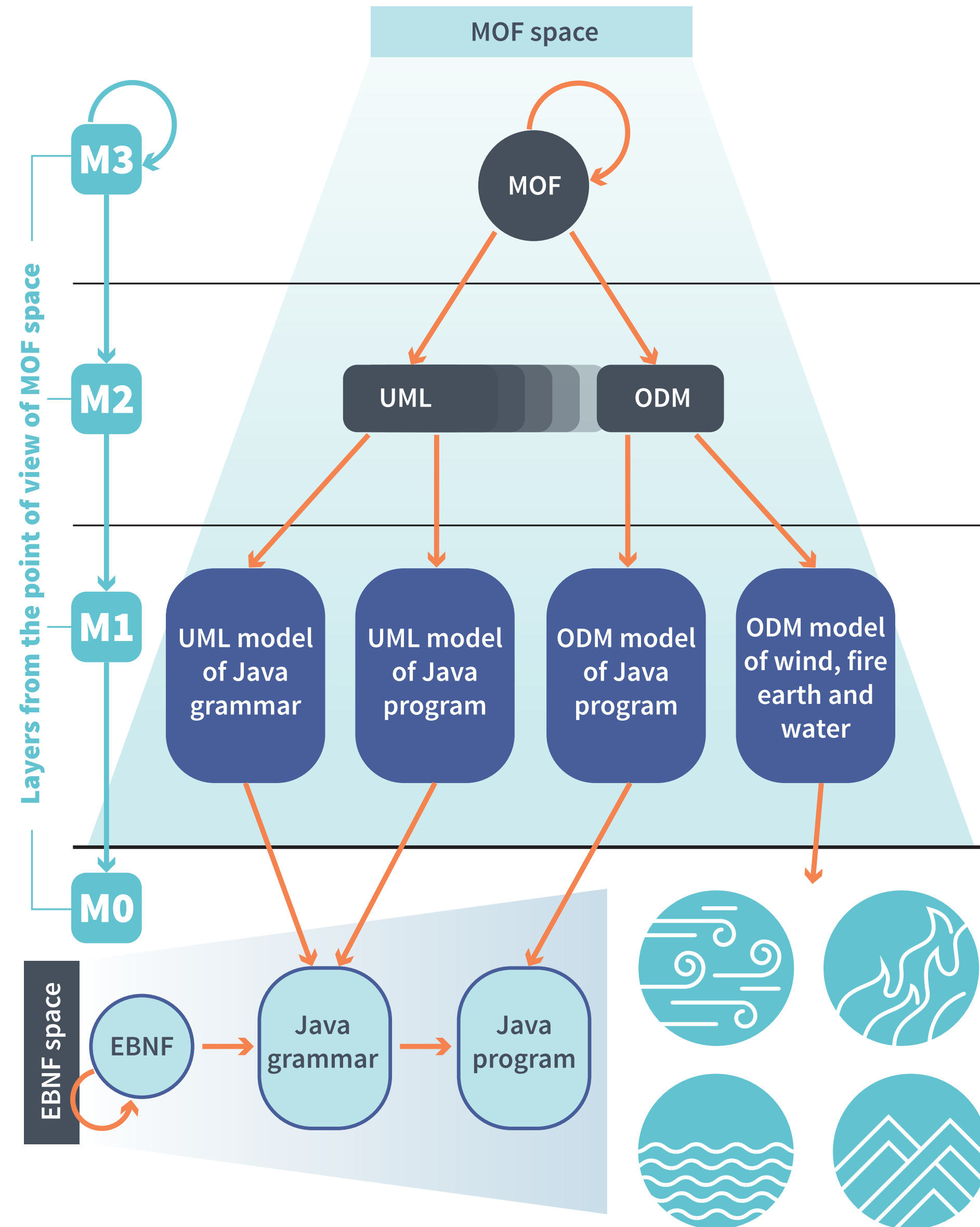
We will talk about low-coding in some more detail when we elaborate on BlackBelt's software engineering technology a little later in this book.

The process of modeling

Let's talk about modeling space. A modeling space is an architecture defined by the meta-model. At its highest level of abstraction, M3, you'll find the Meta Object Facility (MOF). MOF is a standard type system for model-driven engineering. Its aim is to provide an entity system in the Common Object Request Broker Architecture (CORBA), and a set of interfaces, that is used to manipulate and create certain types.

MOF is a section of the Unified Modeling Language (UML) standard, where we can define the modeling language. However, the M3 MOF layer is independent from the UML. The UML is placed on the M2 level, where we can put our own model. This model will contain those entities that we want to represent when the software is running.

This is what a common meta-modeling environment looks like. It consists of four layers. The first layer, the M0 level, is where you will find run-time instances that are used to describe models that represent things from the real world. The second layer, the M1 level, contains the model. The next layer is M2. It is the level that consists of meta-models and UML, where you would define the functionalities of the model in the previous layer. Finally, the M3 level is the realm of the meta-metamodels, a language that we can use to draw up and define the models, the MOF.

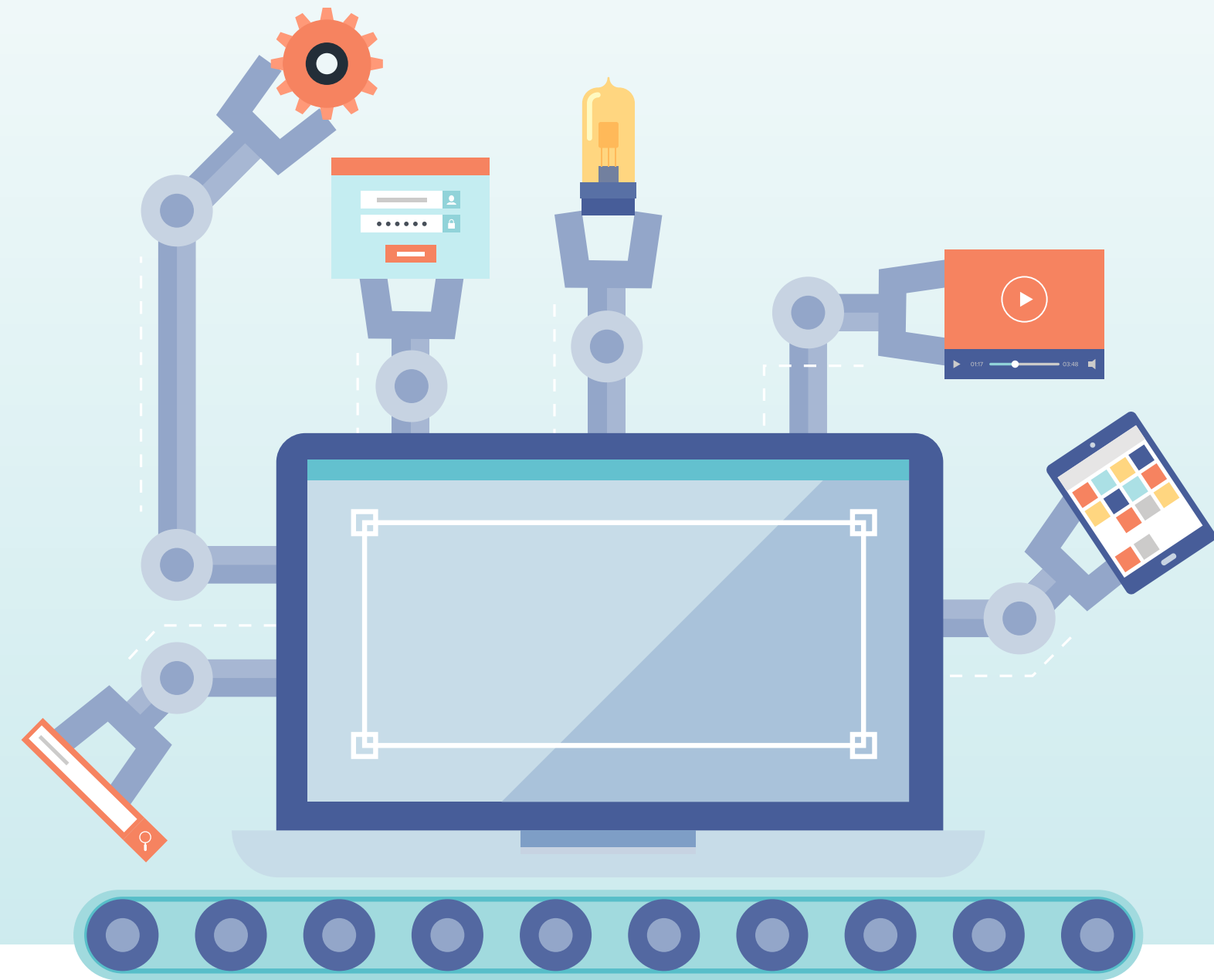


Source: www.jot.fm

The image above shows two parallel, but equal meta-level descriptors. At the top, we find MOF on the M3 level, with the MOF library. As MOF modeling spaces use XML to share their metadata, we can include another space that defines XML. The EBNF (Extended Backus-Naur form) space is another type of modeling space which is visible on the left-hand side on the image. The EBNF space also has a layered structure, although this organization is defined by syntax, not semantics. The EBNF is the metamodel which defines a Java program. Then, it uses the Java program to describe what the entities will look like.

In order to write these meta-model elements, and to use and deploy low-coding platforms, we need to provide interoperability between each meta-layer. This interoperability makes it possible to generate newer and newer models with the help of model conversions.

Model transformation



Model transformations are an automatized way of creating and modifying models. Therefore, we can save energy and time, while we decrease the number of possible errors. Model transformations are about creating interoperability between to meta-models.

UML is not able to describe these modeling structures in a generalized way, so we need model transformations. Why? To answer this question, first we need to know about semantics.

Semantics defines, in a methodological manner, the way a computer follows the programs to be executed in a specific language. In order to express our train of thought and expectations in UML, we should be able to convey semantic content. However, semantics do not have such

capabilities. UML is not for execution, but for modeling, thus its element sequence is not executable. This means that UML encompasses different abstractions and specification techniques, and if we want to use them to deploy a functional system, then we will need to work with an extended, special version of UML called FUMML.

FUMML, or Functional UML, is a subset of UML, which contains the typical structural modeling constructs, like enumerations, data types, associations, and classes. It is also able to model behavior. As a result, a model that has been constructed in FUMML is just as executable as a program written with the use of a traditional programming language.

How BlackBelt creates domain models?

To begin with, we create business terminology where we record the concepts and the relations between them. Then, we define them using the EMF/Ecore pair. EMF stands for Eclipse Modeling Framework. EMF provides the tools and runtime support to create a set of Java classes using a model specification defined in XML. Ecore is the heart of this EMF architecture. It is the second version of a low-coding platform, and it is the MOF layer of the EMF. We use its M3 layer to define modeling languages, and we use the Epsilon Transformation Language to transform different models.

Then on the next level, with the use of a graphical modeling tool, we define the outlook and UI of the model. For this process we use Eclipse's Sirius modeling tool. For the representation, we use displayed elements, shapes, colors and fonts to describe the model. Finally, with the utilization of model driven tools, we can generate, transform, contrast and validate these elements.

JUDO

BlackBelt's project development philosophy

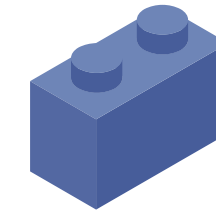
BlackBelt is a software engineering company, which aims to lift the burden of development from their clients' shoulders. We will talk about the company in more detail in the last chapter of this book. At this point, we would like to introduce the unique project development approach of BlackBelt that can help all enterprises and developers who are interested in outsourcing their applications' development.

JUDO is a modeling environment and platform defined and invented by BlackBelt. It is the company's digital business platform serving business objectives. It combines the power of three great modeling surfaces that the company uses to generate more advanced and more modern application platforms.

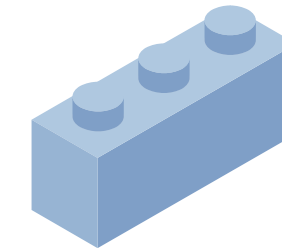
The software engineering process at BlackBelt looks as follows:



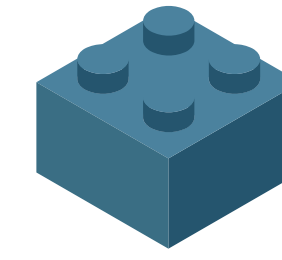
Ideation



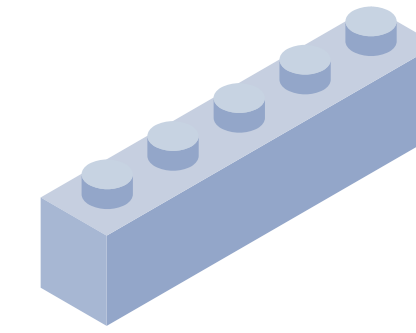
Prototype



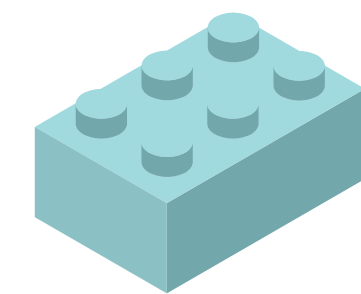
Test



Iteration



Coding



Publishing

By using JUDO, clients can have an insight into the development process while also being able to test the system from time to time. As a result, any new concepts which alter the system from the original concept are feasible due to the fact that JUDO allows immediate modifications. This solution makes the job of the client and the developers easier, as the client can take a look at the software before the finished end-result, therefore the high costs of modifying a complete project can be totally avoided.

With the help of JUDO's low-code platform, it is faster and easier than ever before to generate prototypes and models about the new software. Clients no longer need to worry about stressful and overcomplicated negotiations with drafts drawn on whiteboards or on PowerPoint. Sketching the software's model through JUDO is possible in the blink of an eye.

Low-coding makes the development process a lot faster, as BlackBelt programmers are working with flawless pre-written codes that match each client's expectations. As a result of this innovative technology, clients can see their idea's prototype within a week. At the same time, developing a complete application is possible within several weeks instead of an ever extending period of time.

BlackBelt has the technology and pre-written code that can be used to fill in any prior gaps in systems or environments. Using this highly iterative solution, any app developed by BlackBelt can fit into a company's existing infrastructure.



Why modeling is useful for developers and business individuals?

Modeling and the use of model-driven architectures (MDA) are beneficial for developers and the business domain alike. Let's take a look at its advantages over traditional manual coding.

The advantages of modeling and MDA over traditional manual coding:

- Systems can be specified independently from the supporting platform
- Provides the means to specify the best supporting platform for the app
- Enables the transformation of the software from one platform to another
- Helps to unravel system and environment requirements
- Uncovers hidden misunderstandings and undetermined factors
- Facilitates seamless communication between developers and business individuals
- Business analysts can finally have an insight into the artifacts and models without having programming knowledge
- With the use of model transformation, it is easy to convert one model into another
- Empowers individuals and teams to leverage change and complexity for competitive advantage
- Coupled with low-code platforms, modeling tackles the challenges of modern software engineering

BLACKBELT


Welcome to Blackbelt, a professional application development company

Contact us

 [linkedin.com/company/blackbelt-technology-kft/](https://www.linkedin.com/company/blackbelt-technology-kft/)

 twitter.com/blackbeltech

 [facebook.com/BlackBeltTechnology](https://www.facebook.com/BlackBeltTechnology)

 blackbelt.hu



BLACKBELT is one of Hungary's most dynamically growing software engineering companies. The idea to connect the country's best experts with innovative and high-tech technology was conceived in 2013. The number of employees grew ten-fold within a year, and our exceptional team started to construct the JUDO platform in 2015 and started the beta phase in the same year.

We began to offer our services and technology to foreign companies in 2016 when JUDO was introduced in a client environment. By 2017 our organization matured, and we upgraded our whole enterprise. Last year, in 2018, JUDO has been used on international projects, thus the international market has greatly demonstrated the effectiveness of our methodology.

Today, our superior agile technology enables us to work on numerous international and domestic projects alike. Due to the convenient time zone difference from Hungary's location in Central Europe and our English fluency, we can conveniently distribute our expertise and programming solutions to you. BlackBelt is a smart choice for all Western and Eastern European companies and developers who are seeking a nearshore software engineering company to increase the effectiveness of their business.

If you're looking for a highly skilled and professional company to outsource your software development to, get in touch with us now.