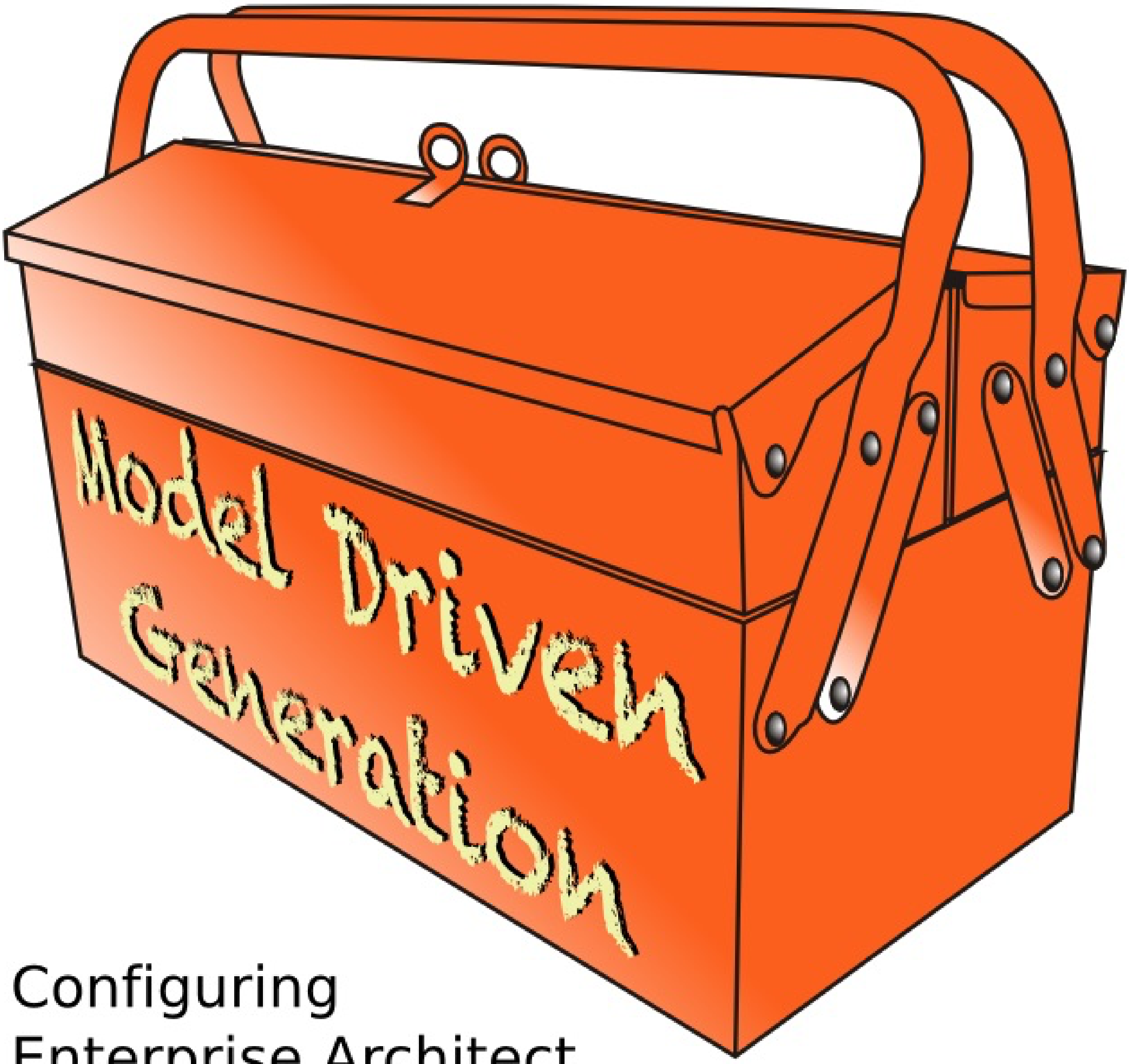
Domain Specific Profiles



Configuring
Enterprise Architect
with MDG Files

By Thomas Kilian

Domain Specific Profiles

Configuring EA with MDG Files

Thomas Kilian

This book is for sale at http://leanpub.com/EA-MDG-Profiles

This version was published on 2021-11-26

Contents

1. Preface

One of the great things in Enterprise Architect is the possibility to extend standard UML modeling by usage of so-called MDG (Model Driven Generation) files. With these you can – in short – add own diagram types offering your own sets of stereotype elements and connectors (with individual shapes) in customized tool boxes. Here is how you can approach the generation of such MDG files.

The EA version used to create this book was actually 12 (build 1215). However, most of the tools are also available in earlier versions of EA. The menu structures change from release to release and you will eventually need to poke around a bit to find the right one. If you really got stuck with the menus or other things just drop me a mail: thomas.kilian@me.com.

Starting with EA Version 15 Sparx introduced a couple of new things with profiles which look more promising than many of the "features" that came during the last years. So this book release catches up with the now recent version 15.2.

2. Copyright and Disclaimer

Also all of the information in this book has been tested by me in many circumstances I can not hold any liability for use of the here presented information[1]. However, I'd be glad to receive any kind of feedback to correct future updates of this book which you will receive for free in turn. Having said this, all information presented here is subject to change without notice.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

[1] I really loathe writing such legal blurb since it should be obvious. By the way: German Law applies! (Does that change anything?)

3. Overview

Having an individual MDG is a must for large projects. Most domains have a specific language that needs to be reflected as good as possible via UML. Whenever you start setting up a MDG you should at least have your domain analysis in a final state. This domain analysis needs to tell you which domain objects exist and which of them are important (need a special optical representation) what diagrams need to be created and in what context they are used.

The following chapters will tell you how to create the domain model and how to derive the MDG from that. Be prepared that those steps need a couple of iterations. That means you will introduce a MDG once you have a scaffold and then need to improve this to make it even more useful. Since changes to a MDG can be crucial I will explain how you can navigate around some of the worst pitfalls.

To avoid a pure hypothetical discussion I will stick to some artificial domain which hopefully can be understood by all readers. It shall be a custom bicycle manufacturer similar to what the guys at Orange County Choppers are doing.

I will describe the single steps one by one so you can create your own MDG at once. It is also possible to open the example model for reference in parallel. Just run a second instance of EA. However, you are encouraged to follow the steps in the next chapter directly. This will give you a quick run through all of the MDG creation process steps. Once you have that skill you can go ahead and learn about more advanced features in the following chapters.

While the first chapters just deal with the MDG surface I try to bring a bit light into what happens behind the scenes. It's nothing you need to know for basic working with MDGs but it often helps understanding why certain things happen the way they do. If you need to know more about that you should have a look into my book Inside EA.

… omitted …

4. Bird's View

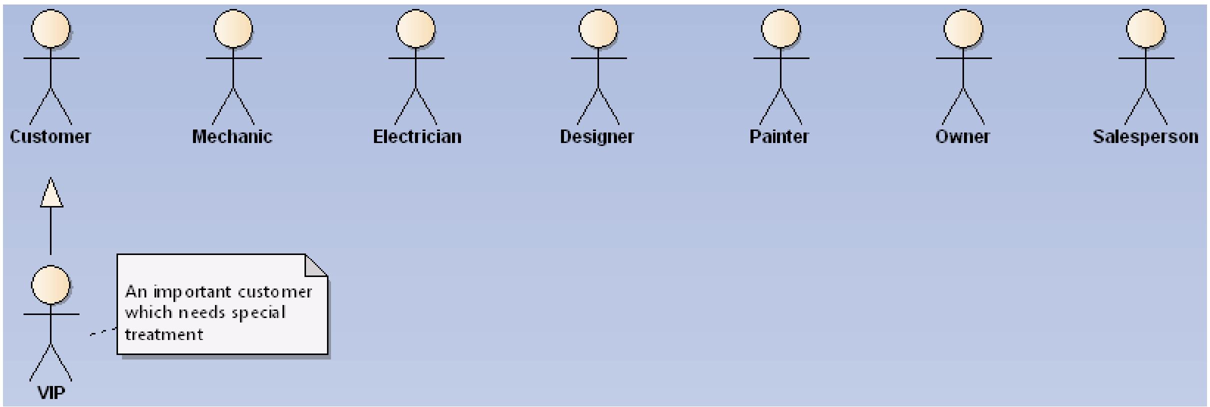
Before you can start creating your MDG it is necessary to do some preparatory work. A MDG shall support development of your UML models and enhance communication with respect to the individual domain you are working in. To actually do that you need to know your domain. And the best way is to sketch a domain model. It's as simple as drawing a class diagram – in theory. In practice it is one of the most difficult tasks since the domain knowledge is usually distributed and well hidden in papers piled in drawers of different staff. Or quite more often just in the brains of the latter.

To get a start with this create a new repository for the MDG design. The model/root node is best called after the domain itself. This can be the industry, company or branch name. I found it best to use Model Driven Architecture (MDA[1]) from the beginning. So the `Views` beneath the model root shall be named `CIM` (computation-independent model), `PIM` (platform-independent model) and `PSM`[2] (platform-specific model). A `Sandbox` view comes in handy for several purposes.

4.1 Stakeholders

The business itself is important. But even more are people doing that business. Only if you know who is involved it will be possible to create something that will be accepted. Your project can only succeed if you get all relevant people aboard. Else prepare for ship wrecking.

Inside the `CIM` create a folder `Stakeholders` with a use case diagram of the same name. Start by putting actors for the most obvious roles onto the diagram. Make sure you write a short description in the notes of each actor. If there is a job description (on paper) just reference that. It is not much work but will help understanding the domain. Not just for you but also for people coming after you that need to get started.



[1] https://en.wikipedia.org/wiki/Model-driven_architecture

[2] If you read my write-up about MDG creation, which formerly had been placed on Sparx' community site and is no located on my own server, you may notice that I put things in `PIM/PSM` rather than `CIM/PIM` as in this book. The reason is that at time of writing the article I had a different focus on abstraction than this time. The issue with MDGs is that you just need a two-tier architecture to describe it while the systems you describe fit best with a three-tier matching the MDA exactly. So it's a bit a matter of taste where to sort in a MDG.

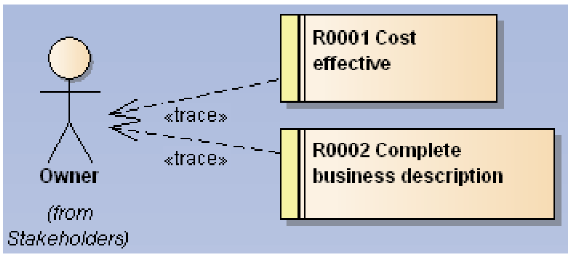
You can augment the diagram by relating the single actors as to the company hierarchy. Use simply dependency relations for that.

The list of stakeholders must not be complete from beginning on. Often you discover new ones in the course of the following process. Just don't forget to add them to the diagram!

4.2 Requirements

Gathering requirements can bring you a hard time. Without the right requirements it is impossible to find the goals you want to reach with modeling. But most times people think that requirements are "obvious" and do not need any effort to be extracted. For the same reason you will not find a budget for this. So to overcome that dilemma you must discipline yourself to write down anything that resembles a requirement. Of course, the best place is this repository where you want to design the MDG.

Place a `Requirements` folder inside `CIM` and create a `Requirements` diagram inside. Eventually you will need some structure for functional and non-functional requirements later. But for now if you happen to find a requirement from any stakeholder just drag the actor as link onto the diagram (you might as well use instances and name the concrete person) create a `Requirement` from the toolbox and `<<trace>>` it to the actor.



It's a good idea to use `Project/Settings/Auto Names and Counters` to define a numbering scheme for requirements. It will not save you from accidentally duplicating number but it's a general help if you do not manipulate the numbering order manually.

4.3 The Domain Model

Once you got the basic set of stakeholders and hopefully a set of requirements you can start analyzing the business domain. Create a folder `Domain Model` with a class diagram of the same name inside the `CIM` view just below the `Stakeholders` package.

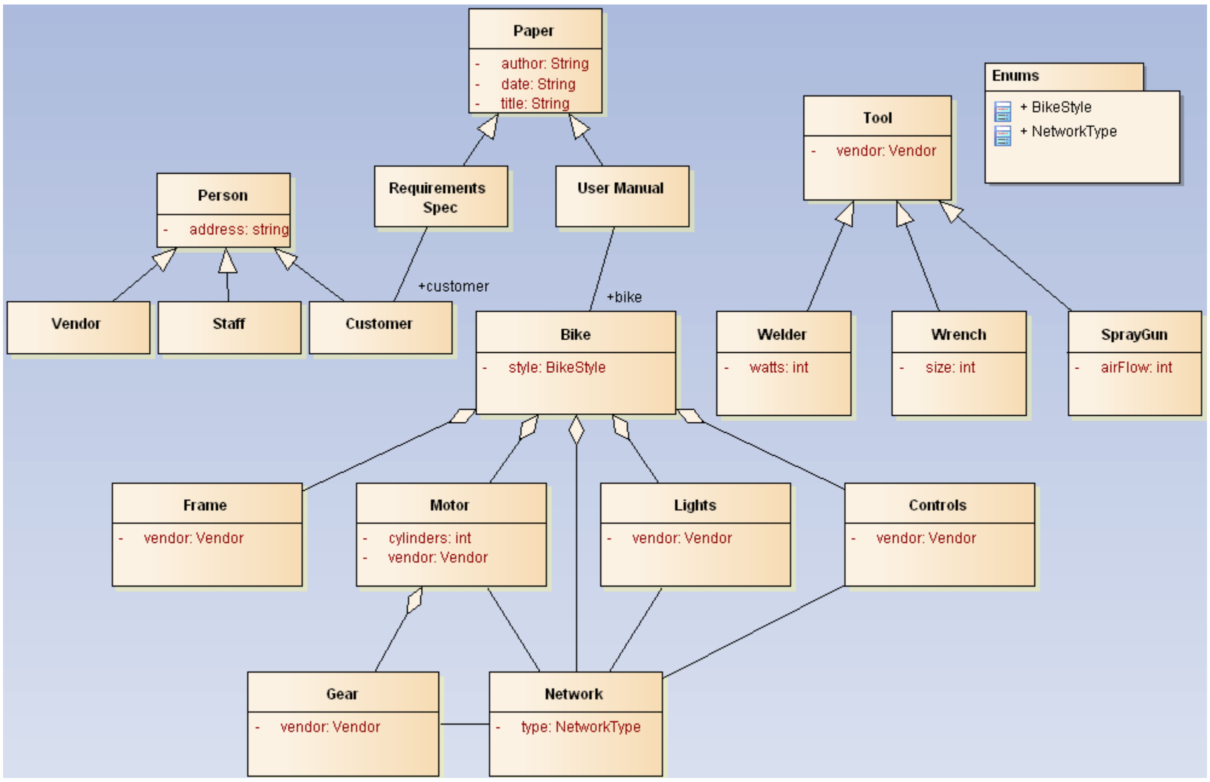
Now start with adding classes for all business objects that come to mind. I sort of mix object and class here. What you actually do is to find the real objects and create a class for it. Each business object shall finally be represented by a class in the domain model.

Of course each domain has very different business objects to deal with. A bank will have accounts and stocks, machine building companies will deal with all sorts of tech paraphernalia, a chemical plant will use chemical material and so on. In a first step just name the single objects and place them on the class diagram. This can well be done as a brain-storming on a white-board.

Eventually the number of domain objects can get large so you need to package them. Papers are a good example to build an own package. Often you can simply identify dependencies between papers and you should draw those directly between the single objects.

The next step is to identify the properties of the single business objects. This is one of the more tricky steps. Finding the properties themselves will often cause you to ask different people and moderate their contradicting answers.

At the same time you need to make up your mind as to what granularity your domain model will go. For example an address can be formalized quite much. But only if your business is limited to a certain geographical area. Once you have more than one area you will find that formalizing address descriptions can easily become a nightmare. It is often easier to start with simple string properties and attach notes to where a later refinement might be useful.



The above model would be some start for our example domain. Definitely it is not complete but from here you will get an easy start and an idea how to continue.

It is important to involve all stakeholders in completing each single business object. But rather than bringing all together in a single work shop (which is often challenging due to time constraints) go and visit one after the other and note what they say. Such big meetings often end up in fruitless discussions and are a simple waste of time. Once you got the different opinions try to unify the diverse ideas and moderate that in smaller workshops. Once you made a picture go ahead and present and discuss this to get a sign-off. Eventually you have to go that round-about once or twice but that should be it.

5. MDG Structure

A MDG file is simply a container for several XML files. Most of them being UML profiles. It is possible to create a MDG with a text (or XML) editor and sometimes you really need to do that[1]. But now we will start the EA-way. And that is by first creating a profile.

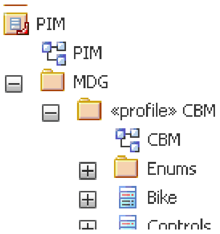
5.1 A Profile

A profile is what contains the elements you define for your domain. A profile as such is meant to keep elements. There are profiles for diagrams, toolboxes and other things as well. In those cases the profile will be attributed with the objective.

Since this chapter is named "A Profile" and not "The Profile" you can guess that there may be multiple profiles. However, a single profile will do in most cases. You would use multiple profiles if you can identify multiple sub-domains which need different treatment. Using multiple profiles is not much more difficult than using a single one. So we start with *the* profile for our example domain.

Copy

The following is *one* way to achieve the desired results. But for me it's *the best* way. It starts with creating a deep copy of the domain model. Just `Ctrl-C` with the focus on the `Domain Model` in the project browser[2]. Go to the `PIM` view, create a folder `MDG`, select that and press `Ctrl-V`. This will create a deep copy of the domain model. Now edit the properties of the copied `Domain Model`. You should rename it to some acronym for the domain since this name will be needed to reference elements which you need to type manually later. In my example it is *CBM* for Custom Bike Manufacturer. Further you must assign the stereotype `<<EAUML::profile>>`[3]. Don't forget to rename the contained diagram as well (with no need to add a stereotype). It should look like this:

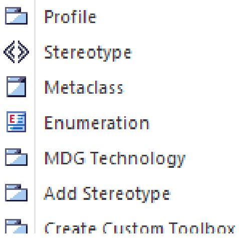


[1] In former EA version you indeed needed to edit the file manually in certain cases. Luckily the need for that has gone in recent EA versions.

[2] I'm not sure since which EA release this is possible. AFAIK it was V10. If it does not work you need to export the package as XMI and import it at the new location with `Strip GUIDs`.

[3] I'm using fully qualilifed names now since the handling of stereotypes got a lot more strict from about V14 of EA. That is every stereotype comes from a specific profile which is the cause for writing `<profile>::<stereotype>` rather than just `<stereotype>`. In former EA versions (and due to less strict definitions in UML) the stereotype had been more or less just an arbitrary string.

Now if you open the diagram and press the space bar you will see that the toolbar shows no longer those of a class diagram but that for a profile.



Cut

The copied domain model still contains all business objects. But often we do not need to model all of them. E.g. the `Tool` is just an abstract classification to have tools inherit the vendor. So it is not an element you would later include in your modeling process.

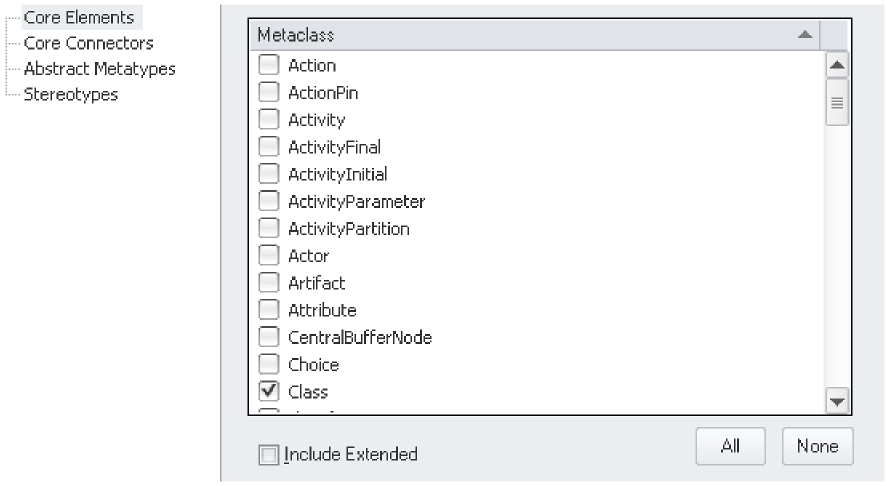
Of course it depends on the domain. In your concrete domain it may well be useful to model those general classes. There is no rule of thumb as to what needs to go into the profile and what can be cut away.

Cutting away those elements will save you some work ahead. Still, if you find a need for those deleted elements later you can create a new copy from the domain model. Instead of cutting elements away you can simply stow them in a package named *Not used* or *Future use*.

Actually the associations created in the domain model will not be used during the MDG creation. But they help identifying structures in the model, so I usually keep them and hide the relations where they are not needed.

Edit

Next create a folder `metaclasses` inside the profile folder. From the new toolbar choose `Metaclass` and inside select `Class`:

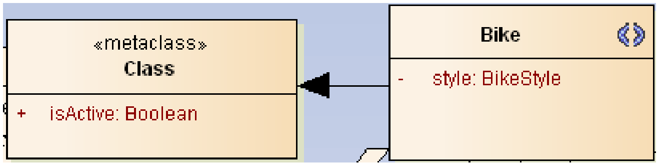


□ Action
□ ActionPin
□ Activity
□ ActivityFinal
□ ActivityInitial
□ ActivityParameter
□ ActivityPartition
□ Actor
□ Artifact
□ Attribute
□ CentralBufferNode
□ Choice
☑ Class

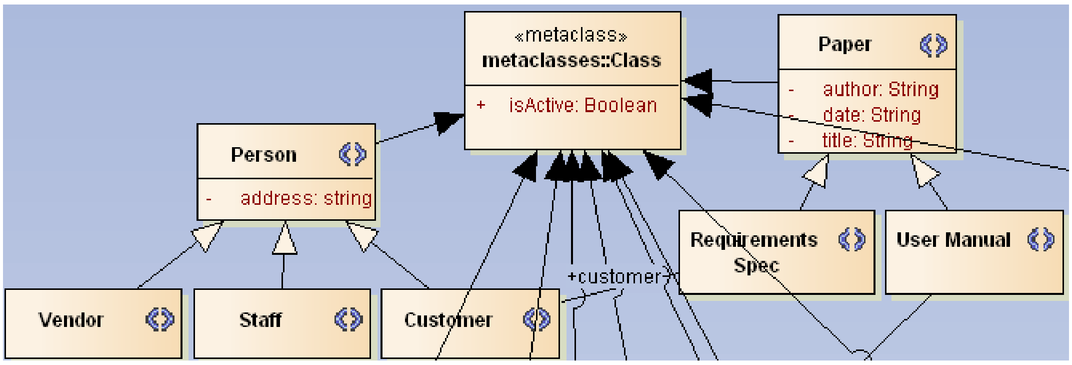
□ Include Extended

This creates a stereotyped class. In the project view drag it into the `metaclasses` folder[4] so it does not clutter the domain objects on top level.

The following steps might be tedious. Start with one of the classes and add a stereotype `<<stereotype>>`[5] manually (without typing the guillemets!). This will add a stereotype icon to the class and the quick-linker will offer an `Extend` relation. Just create that as link to the *Class* metaclass:

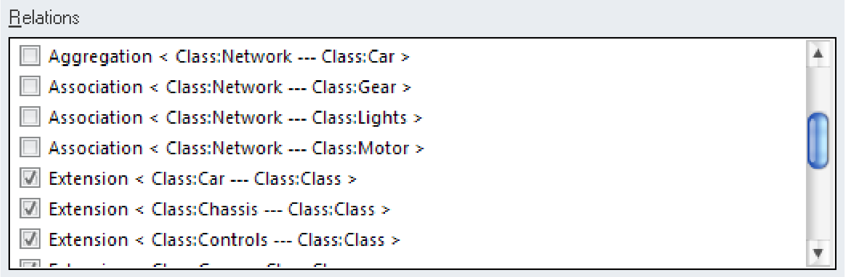


Assuming that all of the domain objects will result in classes continue changing their stereotype and linking them to *Class*. So you end up in some garbled diagram like this



You will be faster if you just paste the text *stereotype* in the `Stereotype` field of the docked properties window after selecting the next element and doing that in a block. Then creating one `Extends` relation from the quick-linker and using `F3` to repeat the relation for the remaining elements will save a lot of clicks.

Finally remove the metaclass from the diagram with `Delete` (not `Ctrl-Del` since that will delete the element completely from the model). You can show the relations in a new diagram named *Class* inside the *metaclasses* package. Drag *Class* onto that diagram and from the context choose `Insert Related Elements`. Now you need to hide all relations except the `Extend`. This is best done opening `Diagram/Visible Relations` and uncheck those to hide:

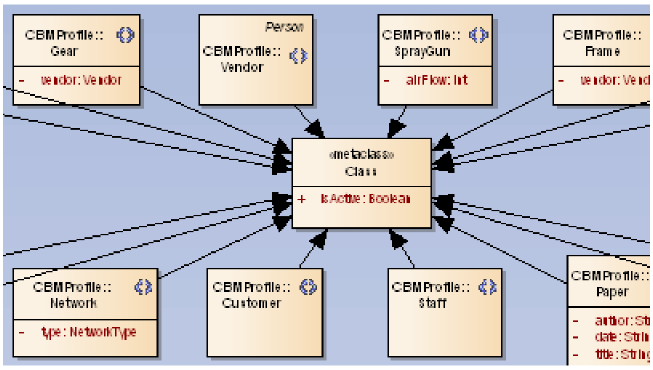

Relations

- □ Aggregation < Class:Network --- Class:Car >
- □ Association < Class:Network --- Class:Gear >
- □ Association < Class:Network --- Class:Lights >
- □ Association < Class:Network --- Class:Motor >
- ☑ Extension < Class:Car --- Class:Class >
- ☑ Extension < Class:Chassis --- Class:Class >
- ☑ Extension < Class:Controls --- Class:Class >

[4] The folder is just for convenience/a matter of taste. It does not matter where the metaclass elements are stored and I like the stereotypes in the package not be cluttered with the related metaclasses. You can as well have both in the same folder.

[5] Actually here you have a stereotype with no profile. Heritage, intention? Currently disucssing with Sparx.

A simple auto-layout (and optional manual work) will then result in something like this:

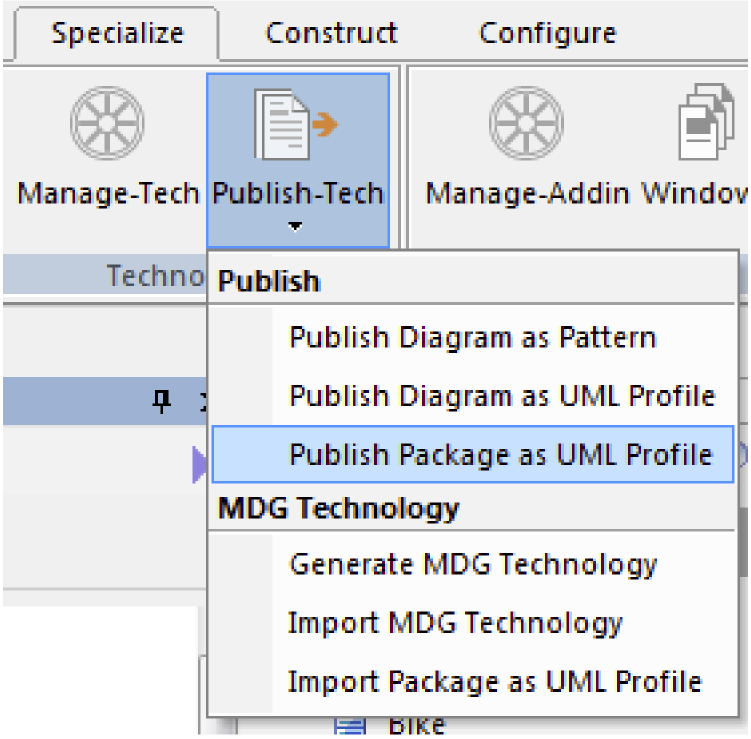


In one of the later EA releases Sparx introduced so-called `Profile Helpers`. In the context menu you will find an entry with that name which might help editing indeed, though it's another set of dialogs you need to learn. Since those helpers make opaque changes as described below I still prefer using the "native" editing. Question of taste, of course. I just get the creeps when things resemble Clippy behavior.

A First Test

At this stage we are ready for a quick test of the new profile. Therefore the profile needs to be exported to a XML file. Select the profile package in the browser and from the ribbon `Specialize/Technologies/Publish-Tech/Publish Package as UML Profile`. Choose an arbitrary file location[6].

[6] In former EA versions the file path was not preserved for a specific profile, but now it is. These go to the dreaded table `t_xref` with the `name` *ProfileOptions*. More about that in my book Inside EA.

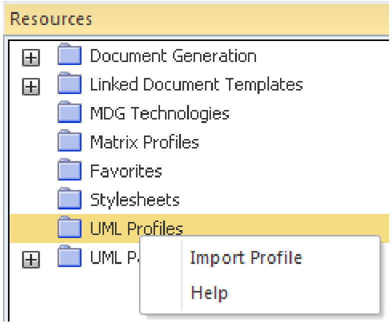


Also you need to give the profile a name which will be `CBM` in our case.

If you look into the file with a text/XML editor you will find all your business objects listed along with some other information. You don't need to handle this file, but knowing how it looks like often helps for debugging or certain tests/quick fixes later.

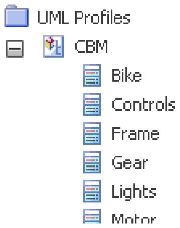
If you left the original relations copied from the domain model you will see a couple of warnings in the EA `System Output` during the profile export. You may safely ignore them.

The next step is to import the profile so it can be used. Open `View/Resources`, from the context of `UML Profiles` choose `Import Profile`



and select the previously exported XML file. Now you can see all the elements that had been defined.

Besides putting the elements in the `Resources` view EA will also put the stereotypes in `Project/Settings/UML Types`. They may conflict here with private stereotypes and can be manipulated. This is different to using a MDG which is entirely held in a separate non-accessible storage area during runtime.



… omitted …

6. Deploying a MDG

A MDG is always loaded temporarily during runtime in some memory which is not directly accessible from an API. That means the stereotypes you had defined in the profiles are not stored in the `t_stereotype` table. Once the MDG is loaded it will however influence the behavior of stereotyped elements.

6.1 Accessing the MDG

Basically there are two recommended and one system reserved variants how to deploy a MDG:

- imported to a single repository and
- shared on a disk or
- in the EA program folder.

⚠ Any way you load a MDG EA will internally distinguish them by their MDG ID. You should in all cases avoid that EA finds the same MDG (ID) in different locations.

ℹ Actually there is one further method to import a MDG: via `Project/MDG Technology Import`. I have only found that from a recent thread on Sparx' forum where one of the Sparxians recommend its use in favor of the `Resources`. I have not yet looked into this but will make an update here soon.

Resource view

In the first case a MDG will be read from disk and stored in EA's `t_document` table per MDG ID. You load the file via the `View/Resources/MDG Technologies`. It will then occur in this view as well as under `Extensions/MDG Technologies` where you find that it shows *Location: Model.* You can enable and disable the MDG here as any other MDG. If you remove it from here it will be removed from the `Resources` (and `t_document`) as well.

To load a new MDG for testing it is sufficient to re-import the file from the `Resources` view.

Shared file

The second way to load a MDG is to tell EA where user MDGs are located. This can be shared folders or URLs. Open `Extensions/MDG Technologies` and click `Advanced`. Here you can add/delete paths and URLs. EA will look into these locations to offer additional MDGs in the `Extensions` dialog. This way you can update the MDG on the fly so users have always the most recent version when starting EA. Vice versa if the file has been erased EA will just disable the MDG and not warn you in any way.

To load a new MDG in this dialog it is necessary to disable it and close the `Extensions` dialog. Now when you re-enable the MDG it will be read from its file location/URL. It is not sufficient to uncheck/tick the MDG without closing.

EA program folder

All of the MDGs listed in the `Extensions` dialog[1] stem from EA's program folder, namely the *MDGTechnologies* sub-folder. You could place your own MDG here, but that is not really recommended. However, it is often a good idea to remove the MDGs you will not use in the course of a configuration management. It might save you some trouble if you get unwanted stereotypes out of the way.

Reloading happens as described above for a shared file.

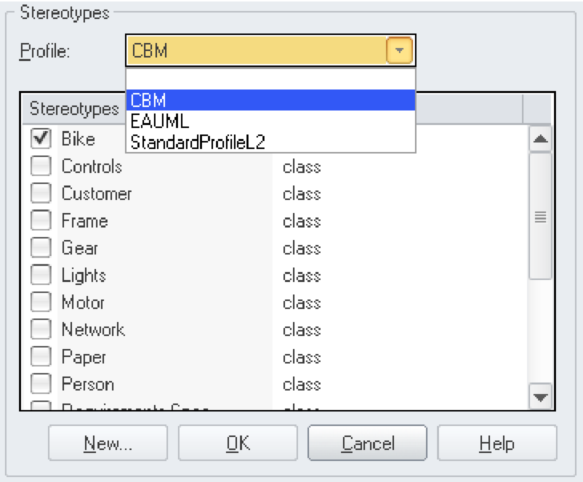
6.2 Using MDG Elements

Loading the MDG is one thing. But using the elements from the MDG is another story. Here I will talk about what happens when you actually use elements from your MDG.

Locating the right elements

Unless you assign elements to according toolboxes of custom diagrams the single stereotypes can be retrieved manually. To do that you can either use the properties window of elements and click the ellipsis button following the `Stereotype` property. Alternatively you can use the `Stereotype` field in the `View/Properties` window and choose `browse other stereotypes...` at the end of the drop down.

[1] After EA has been installed newly.



Stereotypes

Profile: CBM

Stereotypes	
☑ Bike	
□ Controls	class
□ Customer	class
□ Frame	class
□ Gear	class
□ Lights	class
□ Motor	class
□ Network	class
□ Paper	class
□ Person	class

... omitted ...

7. Versioning Profiles

Any MDG will vary over time. It's definitely nothing build from concrete which will last over ages. There are two major reasons for change:

- The domain model changes along with the company itself or because some parts were modeled incorrectly for various reasons.
- The supported process will undergo some optimization not at last because the profile is being used.

So you need to face CRUD scenarios for anything inside a MDG: elements, diagrams and toolboxes.

Let us view the latter: toolboxes. Changes in this area are almost uncritical. It's just the users that might get confused if the toolbox does no longer look like they used to be. There's a simple solution and it's called communication. So if you plan to change toolboxes just be sure to involved all people that may be affected and you are done.

As easy as this was, any other change offers the spectrum from minor trouble to catastrophe. Happily the worst case got more unlikely as it seems that the Sparxians really fixed wrong behavior from previous versions where TV creation from MDGs had changed dramatically. However, the manipulations described below come with absolutely no warranty and are on your own risk! Have a backup! Make a test run in a sandbox! Know what you are doing! A set of bleeding ears is guaranteed anyway.

7.1 Removing Stereotypes

If you happen to remove a stereotype from your profile you need to be sure to do it the right way. First the stereotype might be used in various toolboxes. So make sure to run the `Extended Search`. This will list also attributes which had been named after the stereotype in question. If in our example domain the `Bike` has to be deleted (just hypothetically) the search would list quite a number of elements. Eventually you could limit the search to `<profile>::<stereo>` (i.e. *CBM::Bike* in our domain) or simply `::<stereo>`. Any ghost element in a stereotype would show up as Bike (i.e. with the stereotype icon) in the toolbox.

If you have any elements using the deleted stereotype they will live on as zombies. That means they have an entry in `t_xref`[1] telling they are from your MDG. But for EA your MDG does not have that stereotype and it will not remove the `t_xref` entry. So they appear in the tagged values still under the technology group. To get rid of those zombies you need the SQL[2] pump gun

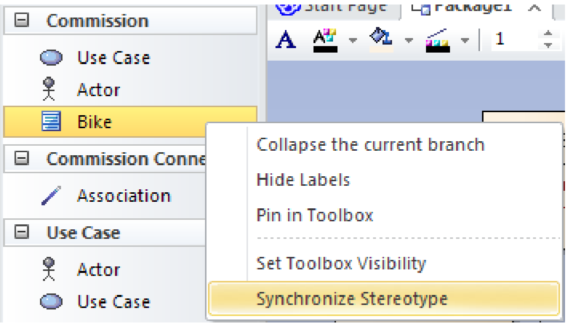
[1] More details about the internal tables are described in my Inside EA book.
[2] Note that the `%` wild card operator must be `*` if you use EAP files.

```
DELETE FROM t_xref WHERE description like "%FQName=<profile>::<deleted>;%"
```

where `<profile>` is the name of your profile and `<deleted>` the name of the deleted stereotype (i.e. in our example *CBM::Bike*).

7.2 Adding an Attribute

This can also be regarded as minor critical. The only thing to do is to synchronize the changed stereotype once the updated MDG is reloaded. You can run the synchronization from the stereotype in `Diagram/Toolbox` from the context menu.



This will add the new TV to where it is needed. The synch will list all elements that have been affected along with the TV that had been added. So far so good. But unfortunately you can not use the list for anything — no copy, not navigation. If you need to rework those updated elements you have to find ways to locate them afterwards.

7.3 Renaming an Attribute

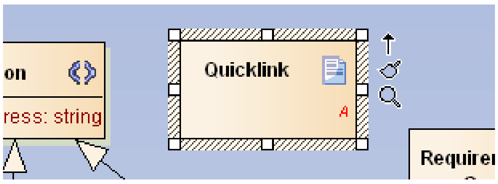
Let's assume you need to rename one of the attributes in a stereotype, be it to correct a typo or for orthogonality reasons. In that case you end up with already created stereotypes still using the old name. Only newly created stereotypes will receive the renamed attribute as TV.

… omitted …

8. The Quick Linker

Here comes another nice gimmick. The quick linker (QL) is a special toolbox which allows context sensitive creation of links and eventually elements created at the other side of the link. This is very powerful for certain model parts. E.g. when creating a line of actions in an activity diagram.

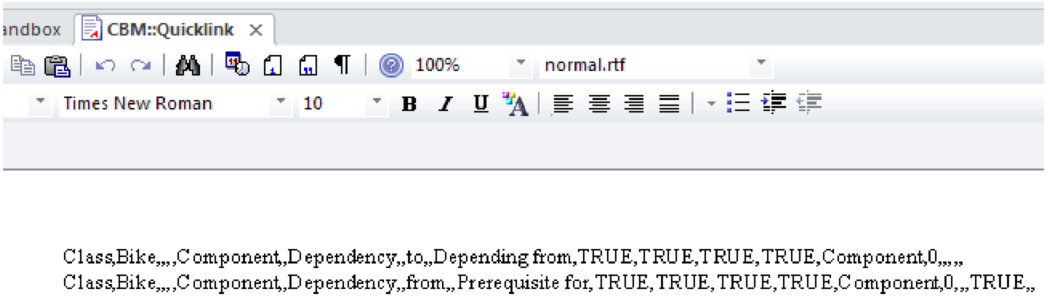
In short the QL is defined by an `Artifact` element in your profile named *QuickLink*. This has a linked document with a CSV formatted text to define the (augmented) behavior of the QL whenever your MDG is active. That far things are easy and you can go ahead creating such an `Artifact`.



Use the `Linked Document` from the context menu and paste the following lines to it

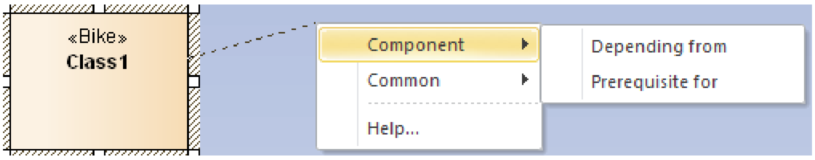
```
1  Class,Bike,,,Component,,Dependency,,to,,Depending from,TRUE,
2  TRUE,,,Component,,,,,
3  Class,Bike,,,Component,,Dependency,,from,,Prerequisite for,
4  TRUE,TRUE,,,Component,,,,,
```

Join lines 1 and 2 and lines 3 and 4 with no space between so you have just two lines. For technical reasons the line breaks in the above example can't be avoided.

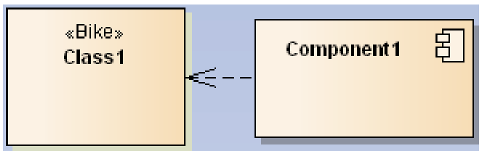


```
Class,Bike,,,Component,,Dependency,,to,,Depending from,TRUE,TRUE,TRUE,TRUE,Component,0,,,,,
Class,Bike,,,Component,,Dependency,,from,,Prerequisite for,TRUE,TRUE,TRUE,TRUE,Component,0,,,TRUE,,
```

Don't forget to save the changes (`Ctrl-S`), export the profile, re-generate the MDG and re-import it for testing: just drag the QL from a `Bike` element you created in the sandbox.



You now have the `Component/Dependency from / Prerequisite for` menu entries for a `Bike` element. Other or not stereotyped elements will not offer this menu. Choosing `Prerequisite for` will create a new `Component`



which has a dependency from the `Bike`.

8.1 Editing the Definitions

Before going into the details I like to explain how to edit the definitions for the QL a bit more efficiently[1]. So useful the QL is in practice so painful (even more) is it to create a custom one for your MDG. Here are a few tips.

As a first step you should change the document format of the linked document to landscape. At least it will help spotting wrong line breaks.

Next add a line before the first you placed in the above example and reuse that for your own QL definitions:

```
//Source Element Type,Source ST filter,Target Element Type,Target ST Filter,
Diagram Filter,New Element Type,New Element ST,New Link Type,New Link ST,
New Link Direction,New Link Caption,New Link + Element Caption,Create Link,
Create Element,Disallow Self connector,Exclusive to ST Filter + No inherit
from metatype,Menu Group,Complexity Level,Target Must Be Parent,Embed element,
Precedes Separator LEAF,Precedes Separator GROUP,DUMMY
```

Also join the above lines so you have a single line of text and precede the definitions in the linked document with it.

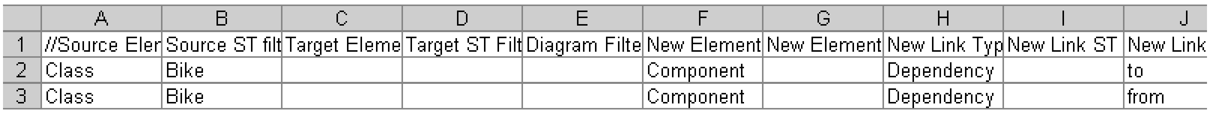
Lines in the definition preceded with `//` are regarded to be comment lines.

To actually edit the definitions it is best to move them to a spreadsheet unless you like to count commas. The RFC for CSV is obviously ignored by anyone (I have not seen a single implementation which worked according to it). So the way to export as CSV and import CSV from file in the spreadsheet is not recommended. Vice versa the CSV export will also likely be scrambled and render to be unusable. I do it in three steps:

[1] Probably it would be a nice idea to create an add-in that allows to manipulate the definitions with a simple form. Anyone to go for that?

- Copy from linked document and paste into `Notepad++`[2].
- Globally change comma to tab chars (using `\t` as replacement).
- Copy the whole text and paste to spreadsheet.

Now you have the data in a more handy format[3].



	A	B	C	D	E	F	G	H	I	J
1	//Source Eler	Source ST filt	Target Eleme	Target ST Filt	Diagram Filte	New Element	New Element	New Link Typ	New Link ST	New Link
2	Class	Bike				Component		Dependency		to
3	Class	Bike				Component		Dependency		from

As you can see the first comment line results in useful headers so you can modify the definitions a little bit easier.

In the examples below I will reference the single columns by the character they have in the spreadsheet. E.g. the column `A` corresponds to `Source Element Type`.

... omitted ...

[2] https://notepad-plus-plus.org

[3] Adjust the column widths as you like.

9. Advanced MDG Techniques

In this chapter I will cover some of the more advanced things you can do with a MDG. Probably most users can live fine with the possibilities that have been explained so far. Alas, you often get appetite while you eat.

9.1 Creating a Shape Script

One really nice thing is that you can adorn the stereotypes from your MDG with an icon. For elements this can be displayed either as shape or iconic in some compass position of the element rectangle. Connectors can also be assigned different shapes. In the first case you should remember that the resulting diagram should not become a bling-bling PowerPoint. So don't overdo this. Create a new shape only for a few important stereotypes and use icons (if at all) for the rest.

… omitted …

10. Testing

First off: there is no debugging assistance. That is: all you see is that your MDG does not work. You even don't get any hint something is rotten. So in any case you want to deliver an updated MDG you need to make sure your changes are ok. Plus it does not have influence on the existing structure.

The most common issue you will find is when seeing a menu like this:

Bike
Gear
Motor
Class

The first you should do is to open the stereotype properties with the ellipsis button and see whether your MDG does appear in the drop down. If it doesn't

- save all profile data and
- re-generate the MDG.

Also one of your toolbox might no longer appear. The most common cause is that the last saved profile went to the wrong location. Try exporting that once again and cross check the filename. Certainly it points to the wrong position. Then just

- save it at the right position,
- re-generate the profile that had been overridden and
- re-generate the MDG.

If it still does not work:

- Check the `MTS` parameters during the generation so the include all and the correct profiles.

Still no luck? Sleep over it. Throw the whole export away (or stow it somewhere) and create everything from scratch. You should be back in business.

Final remark

Congratulations! If you tried out all the steps you should be a pro and even if you skipped various steps you should be able to work on your own and improve your MDG. Personally, I have tried to put all of the information in this book that will be needed from the very beginning to a quite advanced area. The examples provided have been tested to quite some extend but since all this is just from the sandbox it is not comparable to any real life MDG. If you have any issues when trying to follow the steps above or whenever you spot errata please do not hesitate to contact me: thomas.kilian@me.com.

11. Glossary

Actor
An `Actor` is a person or system which interacts with a system in an → `Use Case`.

Attribute
An `Attribute` is an UML element which describes an aspect of a `Class`.

Architecture
A logical structure which describes a system such that from requirements over design to deployment each aspect is covered. Each reader (system users, software engineers, requirements mangers, etc.) of an architecture shall be able to easily extract the part that is relevant for him.

Business Object
An → `Object` that is specifically targeted to some sort of business. A good example is a Customer `Business Object`.

Class
A `Class` is an UML that represents an abstraction of real-world things that are related to each other. A `Class` is build from a classification process. A `Class` usually has different → `Attributes` and → `Operations`.

Connector
A visible → `Relation` between two elements.

Diagram
A visual presentation of part of an UML model. It helps the model reader to understand a certain aspect of the model.

Diagram Toolbox
A context menu for an UML diagram. It contains a relevant subset of UML elements for the individual diagram.

EA Acronym of → Enterprise Architect.

EAUI
Acronym of → Enterprise Architect Unique Interface. A long time ago the Unique Interface was coined by forum member Paolo. I just build the acronym.

Enterprise Architect
Acronym EA. A tool that supports modeling of UML. Developed by Sparx Systems[1], Australia.

Folder
Is another word for → `Package`.

FQN
Acronym for Fully Qualified Name. In this book it refers to the notation `<profileID>::<name>`.

GUID
Acronym for Global Unique IDentifier. A string with hex-chars and dashes which is created in a way such that it is very, very unlikely (but not impossible) to see the same GUID for different elements.

[1] http://www.sparxsystems.com

Instance
Is another word for → `Object`.

MDA
Acronym for Model Driven Architecture. See http://www.omg.org/mda/ and https://en.wikipedia.org/wiki/Model-driven_architecture.

MDG
Acronym for Model Driven Generation. A container format to bundle profiles that can be enabled in EA dynamically.

Method
A piece of code performing some action. In a → `Class` the `Method` usually performs the operation on → `Attributes` of the `Class`. A `Method` can have parameters and a return value.

Object
An `Object` is a placeholder for a real-world thing. `Objects` are instantiated from → `Classes`. Vice versa a `Class` is an abstraction of `Objects`.

Object Oriented
A paradigm that sees the world as → `Objects` abstracted by → `Classes` with → `Attributes` and → `Methods` communicating via → `Messages`.

OO → `Object` oriented.

Operation
A synonym for → `Method`.

Profile
An arbitrary element set based on UML elements. In this book we use the single term `profile` as element set. Additionally the term will be prefixed with `diagram` or `toolbox` for those special subsets.

Package
A `Package` is an element to structure UML models.

Project Browser
An EA window which shows the elements of a repository in a tree structure.

Quick Linker
A tool to connect elements in → EA. Selected elements show a little arrow top right which can be dragged to another element or onto a blank space of a → `Diagram` in order to create a new element along with a → `Connector`.

Relation
An UML concept to tie two elements together. This is expressed through specific → `Connectors`.

RM Acronym for Requirements Management.

Stereotype
A grouping characteristic which in → UML is shown as a string enclosed in guillemets (e.g. `≪device≫`).

Tagged Value
A means to augment UML elements with extra information. Tagged values come in key/value pairs. In → EA the value can be a string, a drop down value or a references to another element.

Toolbox
An icon/name tuple used in menus. `Toolboxes` can be related to diagrams.

Use Case

A `Use Case` is a sequence of `Actions` triggered by an `Actor` and return some measurable value to that `Actor`.

View

A `View` in EA is a special form of a → `Package`.

UML

An acronym for Unified Modeling Language. UML is a language standard published by the Object Modeling Group (OMG). See http://www.omg.org

XMI

Acronym for `XML` Metadata Interchange. Allows to store UML models in XML format for data interchange. Like UML it is a standard published by the Object Modeling Group (OMG). See http://www.omg.org

XML

Acronym for eXtensible Markup Language. It is defined in the XML 1.0 Specification.

YAEAB

Acronym for Yet Another EA Bug. One tends to get sarcastic after many years living with the same set of bugs not being fixed.

12. Bibliography

Gamma, Erich et al.: *Design Patterns*. Addison-Wesley, 1994.

Kilian, Thomas: *Inside Enterprise Architect*. Leanpub, 2012. https://leanpub.com/InsideEA

Kilian, Thomas: *Scripting Enterprise Architect*. Leanpub, 2012. https://leanpub.com/ScriptingEA

Kilian, Thomas: *Shape Script made easy*. Leanpub, 2014. https://leanpub.com/shapescript

Object Management Group (eds.): *Superstructures 2.5*. Internet, 2015. http://www.omg.org/spec/UML/2.5/

Object Management Group (eds.): *UML PROFILE SPECIFICATIONS*. Internet, 2015. http://www.omg.org/spec/#Profile

Object Management Group (eds.): *XMI 2.5.1 Specification*. Internet, 2015. http://www.omg.org/spec/XMI/

Object Management Group (eds.): *XML 1.1 Specification*. Internet, 2003. http://www.omg.org/XML/