

Enabling a tailor-made Modeling Environment for your Enterprise Architecture

By **Thomas Wiman**

Product Manager for MetaModelAgent
thomas.wiman@adocus.com

Abstract

This document gives an example on how to use Eclipse Papyrus and Adocus MetaModelAgent for setting up a domain-specific modeling environment which will support web publishing and model analytics.

The document refers to Enterprise Architecture modeling but the principals described in this document can be applied on any modeling domain.

Content

- Introduction..... 3**
 - The example domain 3
- Section I: Create and Deploy Your Own Domain-Specific Language 4**
 - Step 1: Define a conceptual model of the domain-specific concepts and relations..... 4
 - Step 2: Define the mapping of domain-concepts to UML-items 5
 - Step 3: Define the metamodel to be used by MetaModelAgent..... 7
 - Step 4: Deploy your domain-specific modeling tool and start modeling..... 8
- Section II: Use Your Own Domain-Specific Language..... 9**
 - Model organization 9
 - Domain-specific guidance 9
 - Creating new model elements and relationships..... 10
 - Adding diagrams..... 11
 - Editing existing elements and relationships..... 12
 - Violation Monitoring 12
 - Publishing a model and corresponding guidelines..... 13
- Section III: Analyze Your Own Domain-Specific Models..... 15**
 - Property overviews 15
 - Relationship overviews..... 15
 - Traceability chains 16
 - Dashboards..... 17
- Try it Yourself!..... 18**
- References 19**

Introduction

You are probably working as an enterprise architect, business architect or system architect in a large or mid-size organization. You are in need of establishing a common way to architect, design and document your domain of concern with your already established concepts, terms, rules and constraints.

You really would like to have a modeling tool environment that enforces compliancy with your specific domain, but still compatible with standards, such as UML, for interoperability and flexibility.

You find the modeling tools on the market to be too general or too comprehensive and complex to customize for your own need, or too expensive for your budget. Developing your own tool is certainly not an option.

This paper presents how you easily can achieve a domain-specific modeling tool environment based on Eclipse Papyrus extended with Adocus MetaModelAgent and how it can be used for modeling, model analytics and publishing.

Eclipse Papyrus [1] is an open-source UML-modeling tool based on the Eclipse-platform with extraordinary support for customizations and extensions. *MetaModelAgent* [2] is an extension to Papyrus that makes it possible to define and apply domain-specific modeling without any need of programming. MetaModelAgent also provides model analysis and web-publishing capabilities.

The benefits of using UML as the underlying representation for domain-specific languages is its rich set of elements and relationships with predefined semantics, static and dynamic diagrams for visualizing structure as well as behavior and support for language extensions.

This paper is divided into three sections:

- **Section I** describes how you in a few steps can define your own modeling language and create a modeling tool environment that is customized to your own domain.
- **Section II** demonstrates the domain-specific modeling features you can expect from the tool environment when you have deployed your domain-specific language
- **Section III** presents an overview of the model analysis capabilities in Adocus MetaModelAgent

The complete example used in this article can be downloaded and opened in Eclipse Papyrus extended with MetaModelAgent.

As an alternative to Papyrus, *IBM Rational Software Architect Designer* [3] can be used together with MetaModelAgent with exactly the same functionality and user experience.

The example domain

To not confuse you with a specific kind of architecture, I will use a completely different domain in this paper.

Let's say that you want to capture some information about countries and their major cities and different alternative ways to travel between those cities such as by road, railway, or waterway. For railways and waterways, the available operators are also of interest. I call this domain Cities and Connections.

It is not a typical domain to model, and probably not very useful in practice. But it is a suitable domain for demonstrating the capabilities in the modeling tool environment, as probably anyone can relate to this domain.

I hope that this paper and the example domain will give you an understanding what the proposed tool environment can do for your own domain.

Section I: Create and Deploy Your Own Domain-Specific Language

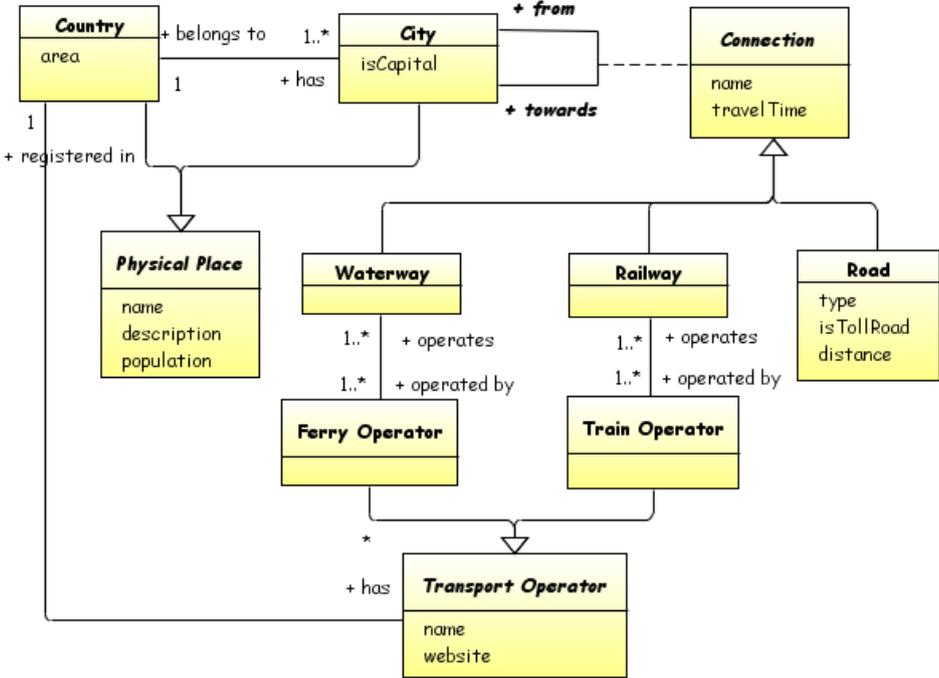
This is about defining a domain-specific modeling language for your modeling domain. This is done in four steps which are described in this section. If your domain is large and complex, I would recommend iterating around these four steps, including deployment and practical use, until the whole domain is covered.

Step 1: Define a conceptual model of the domain-specific concepts and relations

In our example domain, “Cities and Connection”, the following holds: Countries consists of Cities. Cities are connected by Roads, Railways and Waterways and there are Ferry Operators that operates on the Waterways and Train Operators that operates on Railways. Ferry Operators as well as Train Operators are registered in one of the Countries it operates from.

Interesting information to capture is the area and population of Countries, the population of a city and whether a city is a capital or not. Each kind of connection has an estimated travel time and for roads there is of interest to know the distance between the connected cities, the type of road and if it is a toll road or not. For Ferry and Train Operators, the addresses to their websites are of interest.

This will end up in the conceptual model displayed below as a class diagram in standard UML, where some abstract concepts (Physical Place, Connection and Transport Operator) have been introduced to consolidate common characteristics.



Conceptual Model of Cities and Connection Domain

In an architectural context the concept model will contain all your architectural concepts such as Business Process, Activity, Information Item, Organization Unit, System, App-lication, Component,

Subsystem, Module, Interface etc., and all the different kind of relationships that are relevant to keep track of in your architecture.

Step 2: Define the mapping of domain-concepts to UML-items

As all the concepts should be handled in a UML-model, each concept should be mapped to a UML-element or relationship. Preferable, UML-standard semantics should be respected, and the final mapping have to be consistent with UML-semantics.

UML-Profile (Abstract Syntax)

Based on the mapping, a UML-profile [5] is created with a stereotype for each concrete concept, this is also known as *Abstract Syntax*. Icons are assigned to each stereotype to give a better visual cue of the concepts in the UI of the modeling tool.

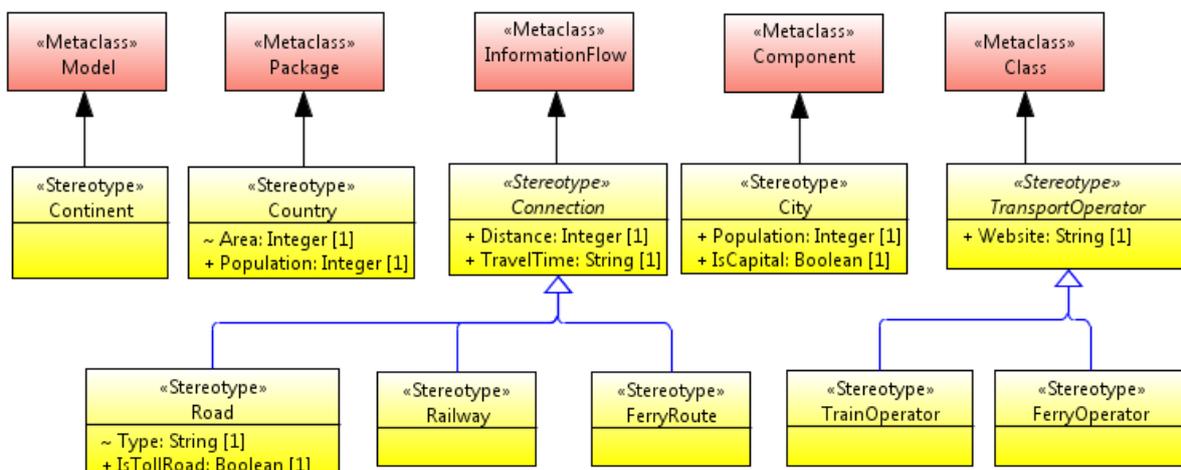
The table below shows how the domain concepts in the example domain are mapped to UML-items (elements and relationships).

Domain concept	UML-item	Stereotype	Icon
Continent	Model	«Continent»	
Country	Package	«Country»	
City	Component	«City»	
Ferry Operator	Class	«FerryOperator»	
Train Operator	Class	«TrainOperator»	
Road	Information Flow	«Road»	
Waterway	Information Flow	«Waterway»	
Railway	Information Flow	«Railway»	

Mapping of domain concepts to UML

You may observe that the concept Continent has been added to the profile to represent the root element in a model.

To capture interesting information about each concept, standard properties on the UML-items are pre-ferred. If standard properties are not sufficient, domain-specific properties are defined by adding attributes to the stereotypes in the profile. The final UML-profile for the Cities & Connections domain looks like this:



UML Profile

You may notice that in the UML-profile above, there is no property defined for holding the reference from a Waterway to Ferry Operators, neither from a Railway to Train Operators, the reason is that Information flows has a standard property named Conveyed which has been decided to be good enough for holding this kind of references.

Diagram CSS-Stylesheet (Concrete Syntax)

In Eclipse Papyrus, the appearance of elements and relationships in a diagram, also known as *concrete syntax*, can be controlled by cascading style sheet (CSS) files [6], analogue with styling web pages using CSS-files. For that reason, a CSS-stylesheet has been developed for the Cities & Connections domain, here is a small excerpt from that stylesheet:

```

Class, Component {
  fontColor: black;
  bold: true;
  isNameWrap: true;
  displayIcon: true;
  gradient: none;
}

Component > Label[type=StereotypeLabel]{
  visible: false;
}

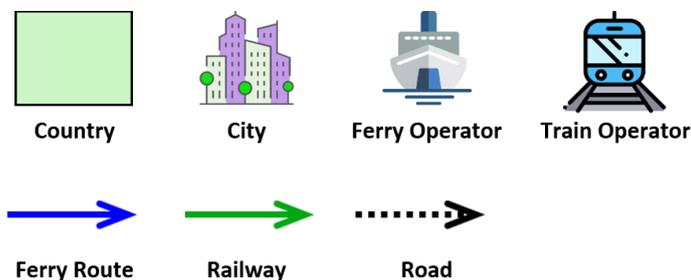
Component > Compartment[kind="symbol"] {
  visible: true;
}

[appliedStereotypes~=City] {
  transparency: 100;
  followSVGSymbol: true;
  displayIcon: false;
  svgFile: 'platform:/resource/MMA Demo Cities And Connections/Graphics/City.svg';
  floatingLabelOffsetWidth: +0;
  floatingLabelOffsetHeight: +100;
  displayName: false;
}

```

CSS Stylesheet

As seen in the CSS-file above, a stylesheet can refer to vector-based graphics in SVG-files to be used as element symbols in diagrams, overriding standard UML-symbols. For our domain SVG-graphics have been added to represents Cities, Train Operators and Ferry Operators:



SVG Graphics

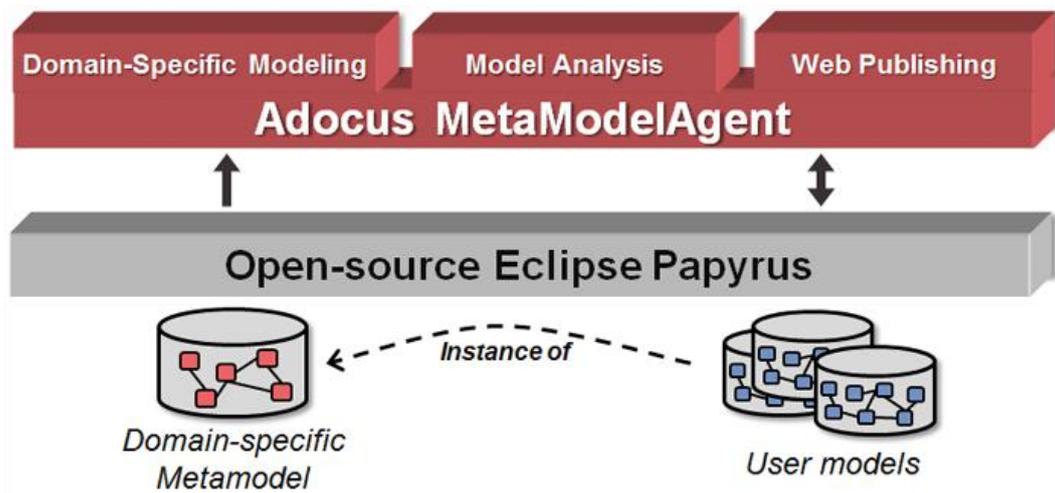
Step 3: Define the metamodel to be used by MetaModelAgent

So far, we have defined a UML-profile with stereotypes and stereotype attributes representing domain-specific concepts and interesting information, with associated icons and a CSS-stylesheet for diagram representation. This is actually all artifacts needed to model our domain in Eclipse Papyrus.

But what about constraining the ways our domain models can be constructed and ensuring compliancy to the domain-specific language? Of course, you can add some complex OCL-constraints to your stereotypes and run some batch validation. But it would be much better if the tool's UI could enforce a correct usage of all the concepts and guide the user to create correct domain-specific models.

This is where MetaModelAgent from Adocus comes to play. MetaModelAgent provides a UML-based metamodel notation [7] where the one that is responsible for the domain easily can define all the rules and constraints that are related to the domain.

Based on the metamodel, MetaModelAgent can then, without need for any programming, provide a domain-specific UI to the user. The UI will include domain-specific menus, views, wizards, and tool palettes that supports the user in creating domain-specific models that are compliant to all the rules and constraints.



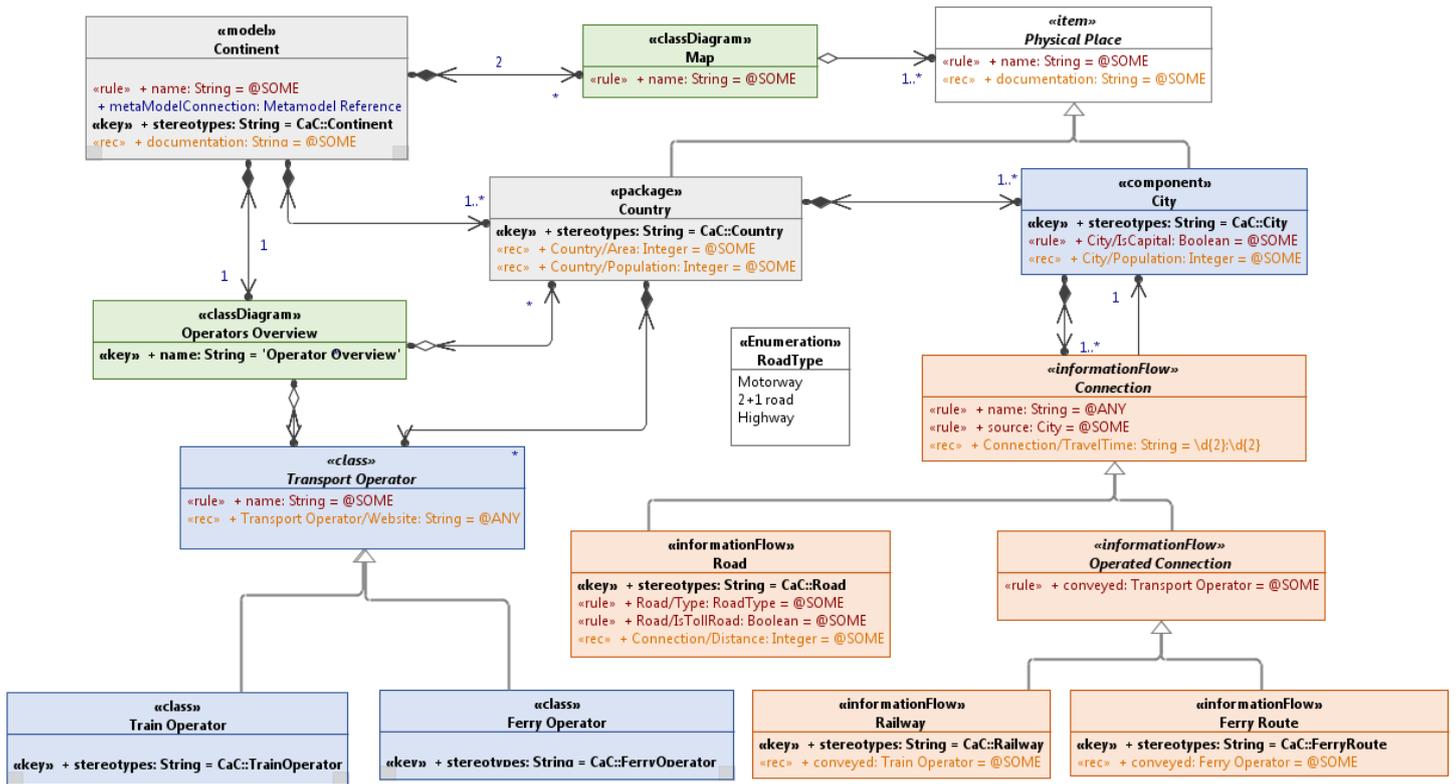
The concept of MetaModelAgent

The metamodel can be regarded as a refined and extended conceptual model:

- A **Class** represent a metaclass and defines the kind of UML-element to use for a specific concept.
- A **Class Attribute** represents a significant property for the concept represented by the metaclass and defines constraints on the property's value.
- A **Composite Aggregation relationship** between metaclasses defines the valid model structure.

The metamodel notation supported by MetaModelAgent is very comprehensive and can capture almost any kind of rules and constraints that holds in a domain.

Below you have an excerpt of the metamodel for the Cities and Connectors domain.



Metamodel of Cities & Connections domain-specific language

Besides defining all rules and constraint of your domain-specific modeling language in the metamodel, you can also add guidance in terms of textual documentation to all metaclasses and metaclass attributes. The guidance will then appear in a guidance view, wherever you select an item or a property anywhere in the UI.

Step 4: Deploy your domain-specific modeling tool and start modeling

There are at least two alternatives to deploy your domain-specific modeling language (UML-profile, CSS-stylesheet, Graphics and Metamodel) in Papyrus.

- All the artifacts can be put into an Eclipse Project that is imported to each user's workspace. This is the preferred variant during testing and with few users as it is very agile where changes can be made without restarting the tool session.
- All the artifacts can be included in an Eclipse-plugin that is referred by an Eclipse Feature and provided as an update site for all users to install. This is the preferred variant when the included artifacts are stable and there are a lot of users. How to deploy your domain-specific language in a plugin, feature and update site is outside the scope for this article. Details can be found in ref [8].

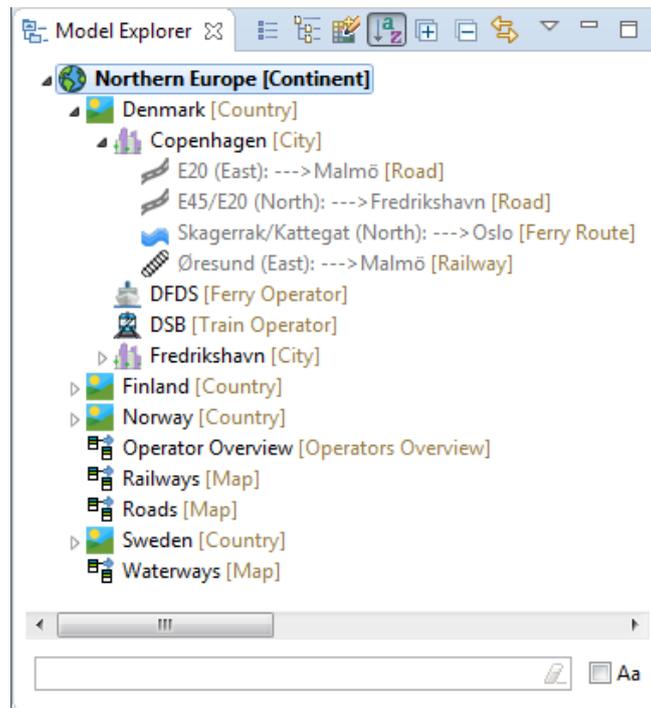
Section II: Use Your Own Domain-Specific Language

To demonstrate the “Cities and Connection” modeling language, an example model has been developed which covers major cities and connections in Northern Europe. The example model is not exhaustive at all and there is no guarantee that the information within the model is correct.

In this section you will find screenshots of how this example model looks like in Eclipse Papyrus ex-tend-ed with Adocus MetaModelAgent.

Model organization

The *Papyrus Model Explorer* will display all content of the model with domain-specific icons and the concept terms as suffix.

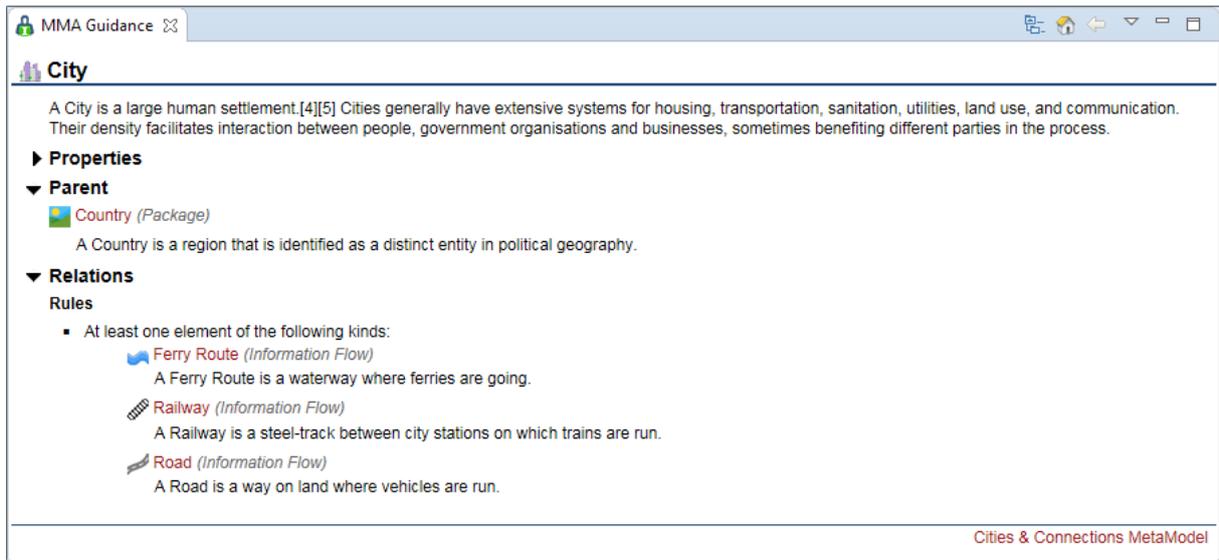


Papyrus Model Explorer

In an architectural context the Model Explorer will display the organization and decomposition of all architectural elements and their relationships.

Domain-specific guidance

The *MetaModelAgent Guidance View* will explain all domain-specific concepts selected in the UI. The explanations displayed have been added as metaclass and metaclass attribute documentation in the metamodel.

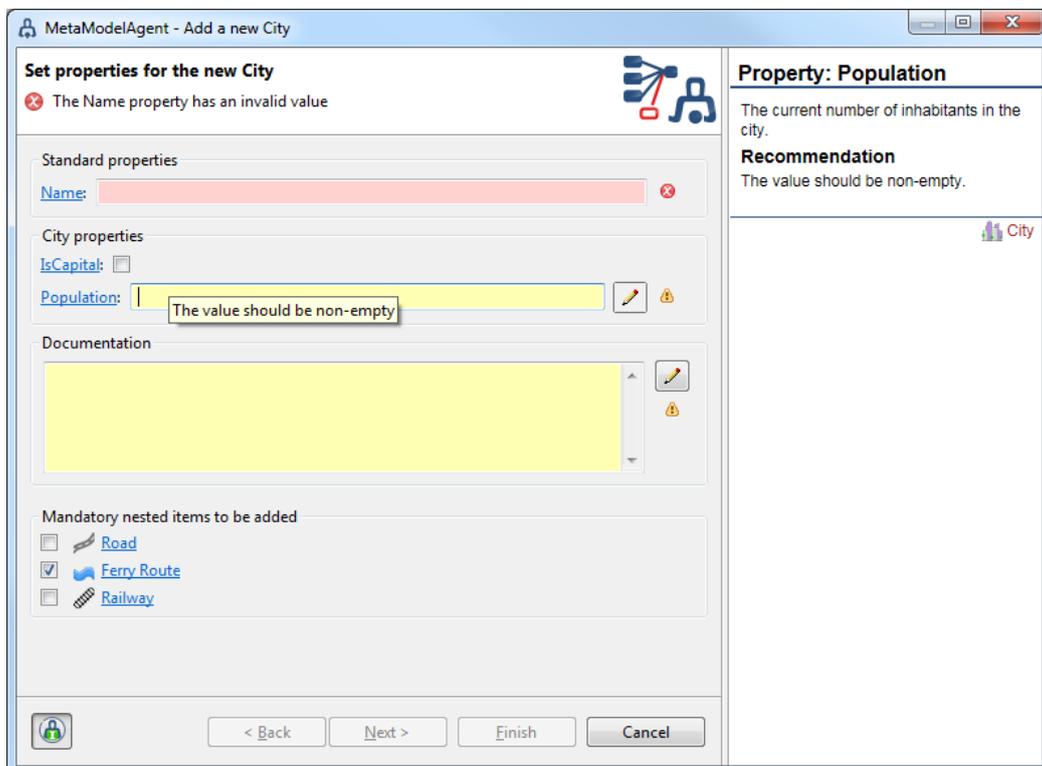


MetaModelAgent Guidance View

The quality of the guidance will depend on the effort put into describing each concept. If you have architectural concepts that are new or complex for the modelers, you should put extra attention in providing good explanations. As this view is an embedded web-browser the guidance can embed any kind of multimedia such as pictures, sound and videos.

Creating new model elements and relationships

New elements can be created from a domain-specific context menu in the Model Explorer View or by using the domain-specific section of the diagram editor palette. Relationships are added using the domain-specific context menu in the diagram editor. In all cases, the **MetaModelAgent Add Wizard** will pop-up to guide the user in adding correct property values for the element or relationship.



MetaModelAgent Add Wizard

The Add Wizard will be unique for each concept, only focusing on those significant properties that have been defined in the UML-profile and in the metamodel. By providing live validation of entered values, the model can be sure to create items that are consistent with the metamodel.

Adding diagrams

In the *Papyrus Diagram editor*, you will be able to visualize your concepts and relations styled by the CSS-stylesheet.

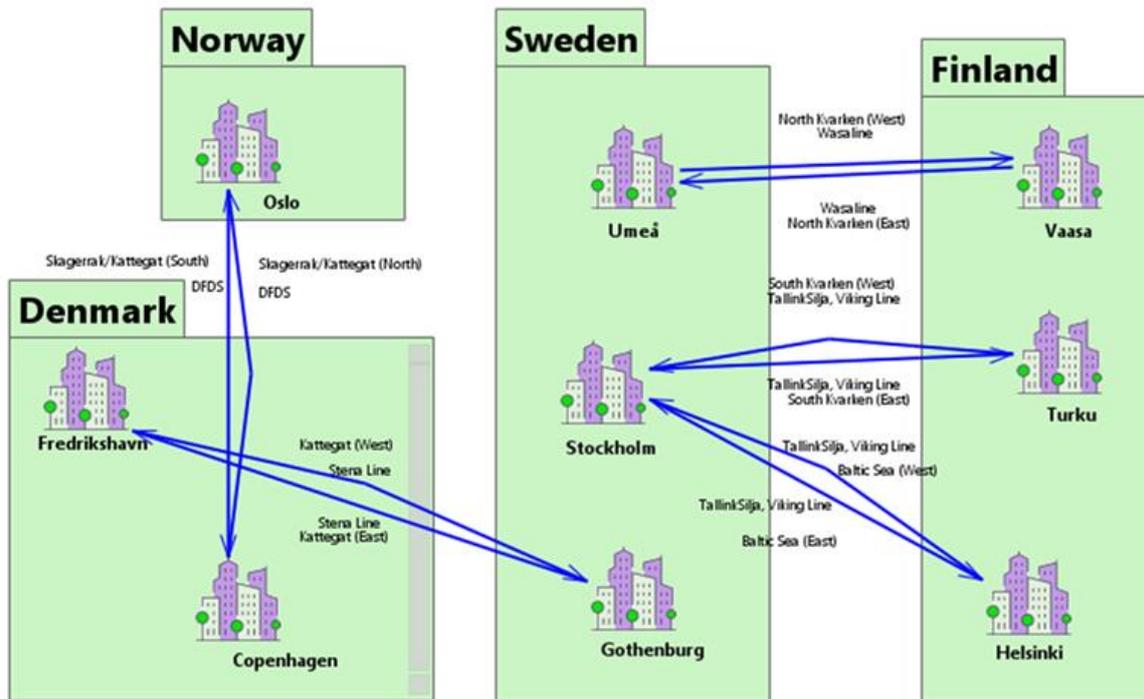


Diagram of countries, cities, and connections

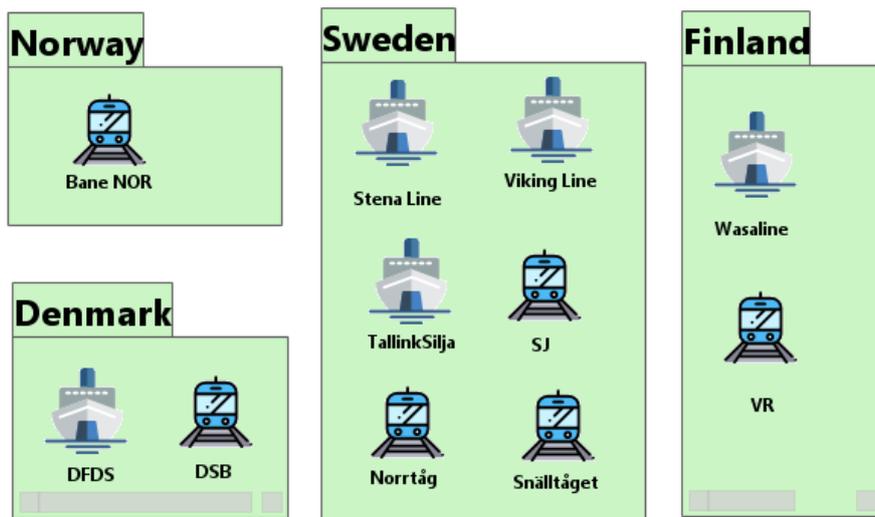
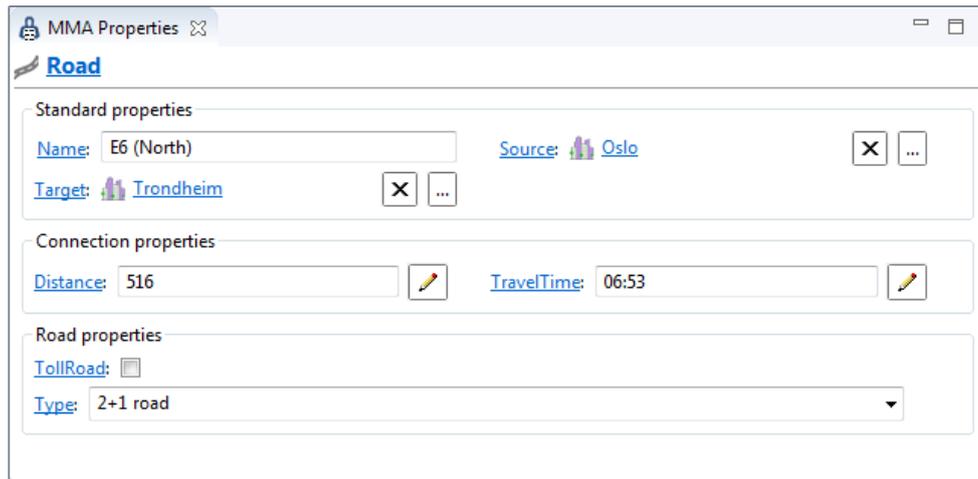


Diagram of Countries and Travel operators

In an architectural context, these kinds of diagrams can be used to visual the architectural decomposition and at the same time display significant relationships between architectural building blocks. UML sequence diagrams, activity diagrams and state-machine diagrams can be used to visualize architectural behavior. Normally several diagram will be needed to express different viewpoints of the architecture.

Editing existing elements and relationships

The *MetaModelAgent Property view* will display only significant properties as defined in the meta-model for the element or relationship selected in the UI. Property values are validated as they are typed in, against defined constraints in the metamodel.



MMA Properties

Road

Standard properties

Name: E6 (North) Source: Oslo

Target: Trondheim

Connection properties

Distance: 516 TravelTime: 06:53

Road properties

TollRoad:

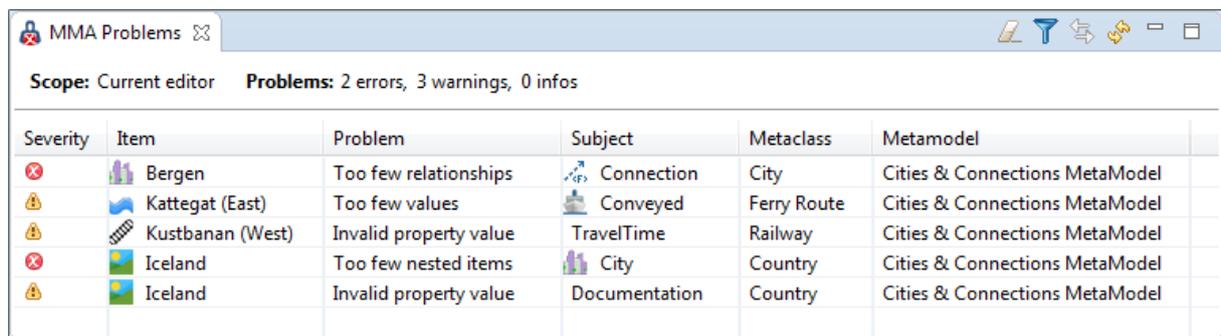
Type: 2+1 road

MetaModelAgent Property View

When editing already created items you will have the same kind of support as in the Add Wizard to enter valid property values only. Any other value will be indicated by colored background and decoration indicating the severity.

Violation Monitoring

Even when using domain-specific wizards and property view, you may end up in incorrect or incomplete constructions. The *MetaModelAgent Problem View* will highlight all outstanding violation against the metamodel in a spreadsheet view that also provides quick fixes whenever possible.



MMA Problems

Scope: Current editor Problems: 2 errors, 3 warnings, 0 infos

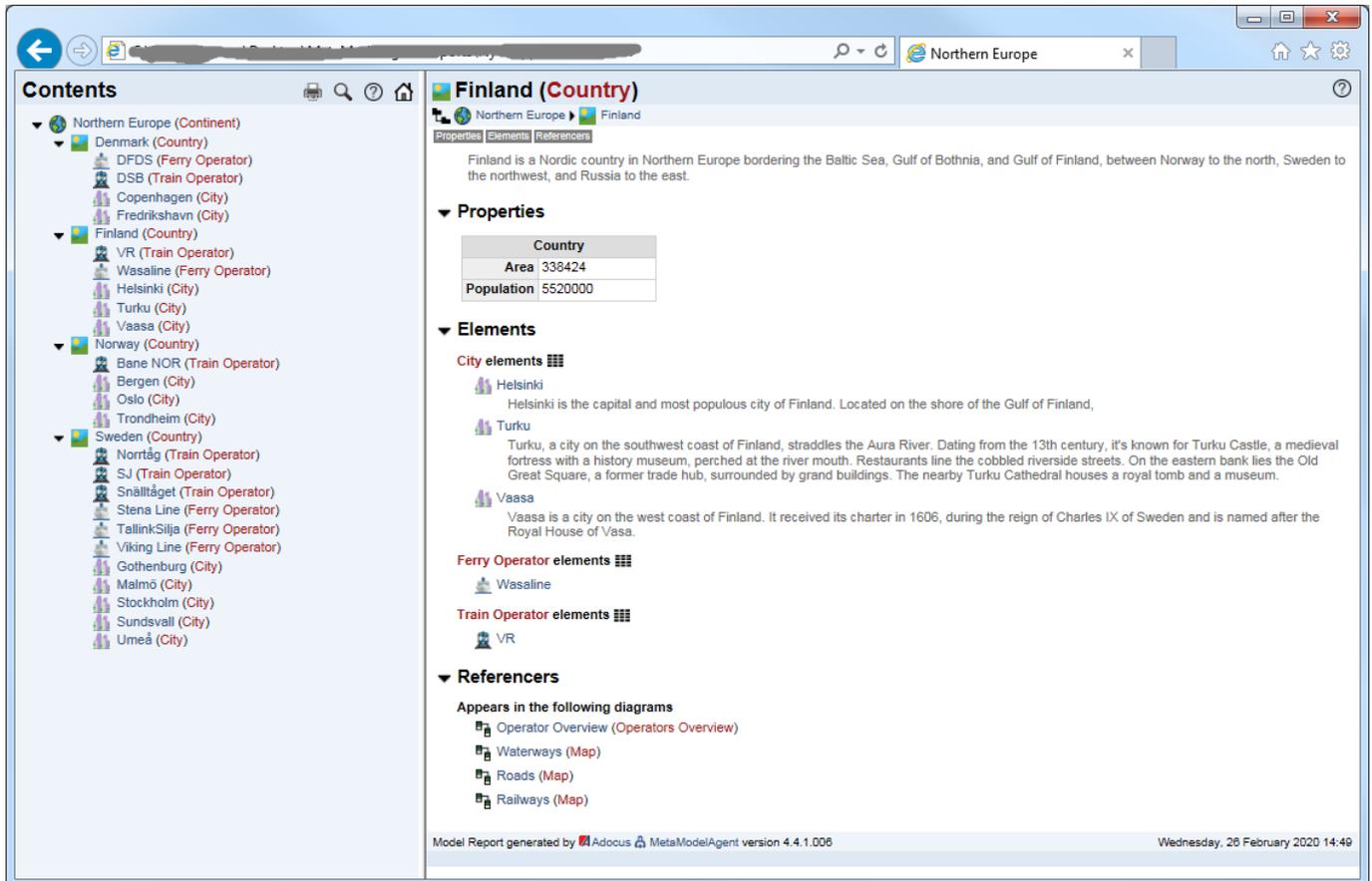
Severity	Item	Problem	Subject	Metaclass	Metamodel
✖	Bergen	Too few relationships	Connection	City	Cities & Connections MetaModel
⚠	Kattegat (East)	Too few values	Conveyed	Ferry Route	Cities & Connections MetaModel
⚠	Kustbanan (West)	Invalid property value	TravelTime	Railway	Cities & Connections MetaModel
✖	Iceland	Too few nested items	City	Country	Cities & Connections MetaModel
⚠	Iceland	Invalid property value	Documentation	Country	Cities & Connections MetaModel

MetaModelAgent Problem View

Publishing a model and corresponding guidelines

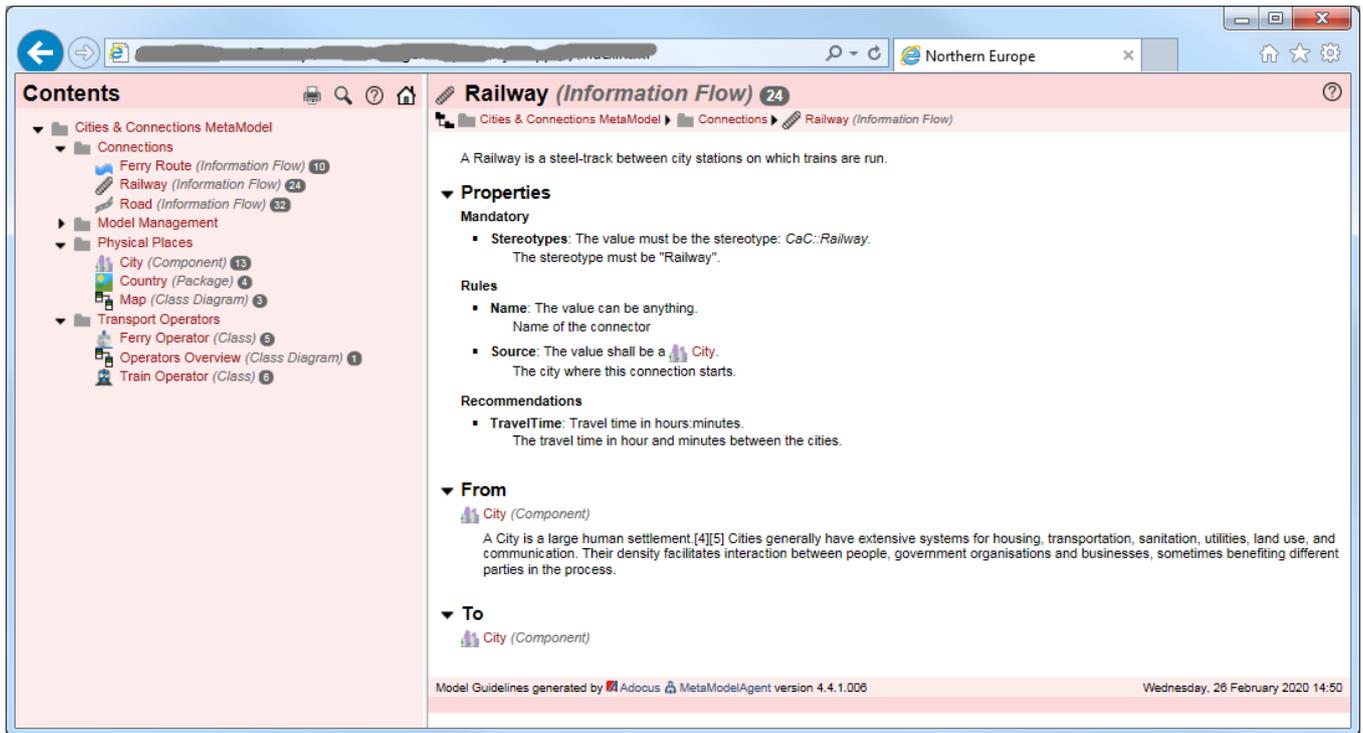
A model is often consumed by stakeholders not directly involved in the modeling task. They will there-for normally not have access to the modeling tool: Even if they have access, the modeling tool may be too complex to use for their needs. To be able to consume a model in a simpler way is an important requirement from them.

MetaModelAgent comes with a *web site generator* that can generate a stand-alone static *model report website* of one or several models with some unique cross-reference facilities and search capabilities.



Web published Model Report

Besides publishing a web site of user models. MetaModelAgent can also publish a *guideline report website* of the metamodel holding the definition of the domain-specific language. That website could be seen as a reference manual of the domain-specific language and will reduce the needed for additional language documentation.



Web-published Guideline report based on metamodel

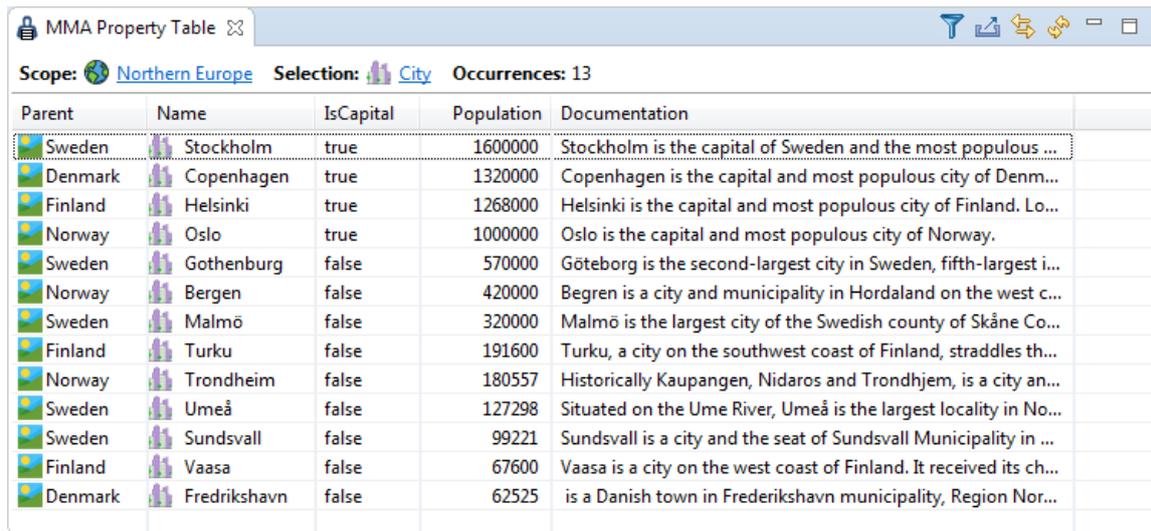
If you choose, both the model report website and its corresponding guideline report website can be integrated with each other making the concept definition being available within a mouse-click when navigating the elements in the model.

Section III: Analyze Your Own Domain-Specific Models

Beside the domain-specific modeling capabilities that are provided by Eclipse Papyrus together with Adocus MetaModelAgent, MetaModelAgent also provides some advanced model analytics features [10] where several of them are never seen before in a modeling tool.

Property overviews

The interactive *MetaModelAgent Property Table View* will display all occurrences of a selected concept together with values of all significant properties in an editable spreadsheet layout. This view also provides bulk editing and export to comma-separated files.



The screenshot shows the 'MMA Property Table' window. The title bar includes a search icon, a refresh icon, and a close icon. Below the title bar, the window displays the following information: Scope: Northern Europe, Selection: City, Occurrences: 13. The main content is a table with the following columns: Parent, Name, IsCapital, Population, and Documentation. The table contains 13 rows of data, each representing a city in Northern Europe.

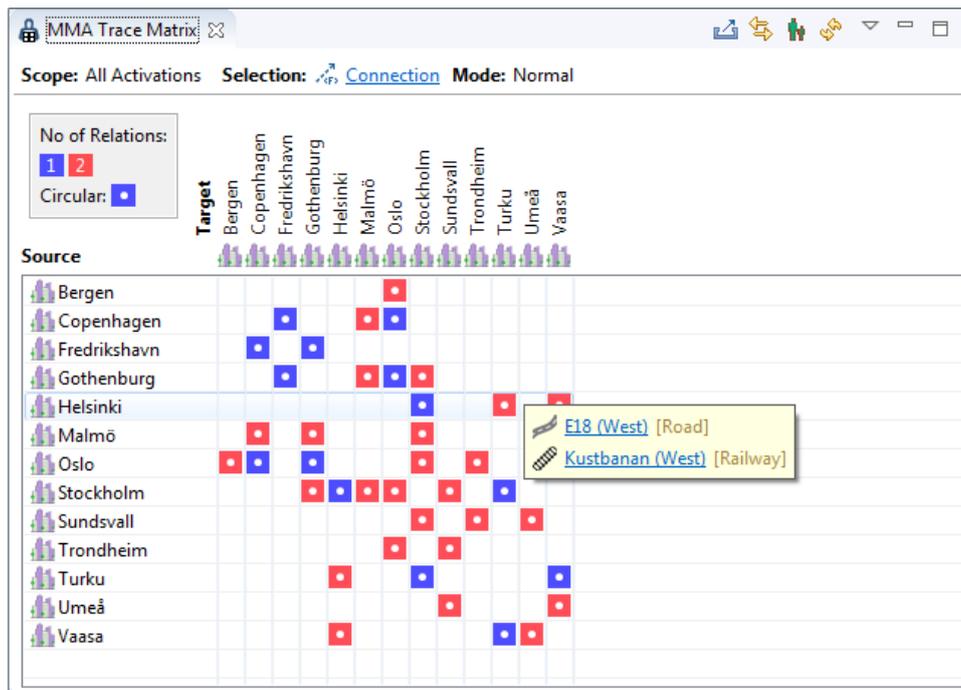
Parent	Name	IsCapital	Population	Documentation
Sweden	Stockholm	true	1600000	Stockholm is the capital of Sweden and the most populous ...
Denmark	Copenhagen	true	1320000	Copenhagen is the capital and most populous city of Denm...
Finland	Helsinki	true	1268000	Helsinki is the capital and most populous city of Finland. Lo...
Norway	Oslo	true	1000000	Oslo is the capital and most populous city of Norway.
Sweden	Gothenburg	false	570000	Göteborg is the second-largest city in Sweden, fifth-largest i...
Norway	Bergen	false	420000	Begren is a city and municipality in Hordaland on the west c...
Sweden	Malmö	false	320000	Malmö is the largest city of the Swedish county of Skåne Co...
Finland	Turku	false	191600	Turku, a city on the southwest coast of Finland, straddles th...
Norway	Trondheim	false	180557	Historically Kaupangen, Nidaros and Trondhjem, is a city an...
Sweden	Umeå	false	127298	Situated on the Ume River, Umeå is the largest locality in No...
Sweden	Sundsvall	false	99221	Sundsvall is a city and the seat of Sundsvall Municipality in ...
Finland	Vaasa	false	67600	Vaasa is a city on the west coast of Finland. It received its ch...
Denmark	Fredrikshavn	false	62525	is a Danish town in Frederikshavn municipality, Region Nor...

MetaModelAgent Property Table View

In an architectural context, the Property Table View will make it really easy to get an overview of all properties for all archi-tectural elements of the same kind. The export capabilities make it easy to transfer data to Excel for further post-processing.

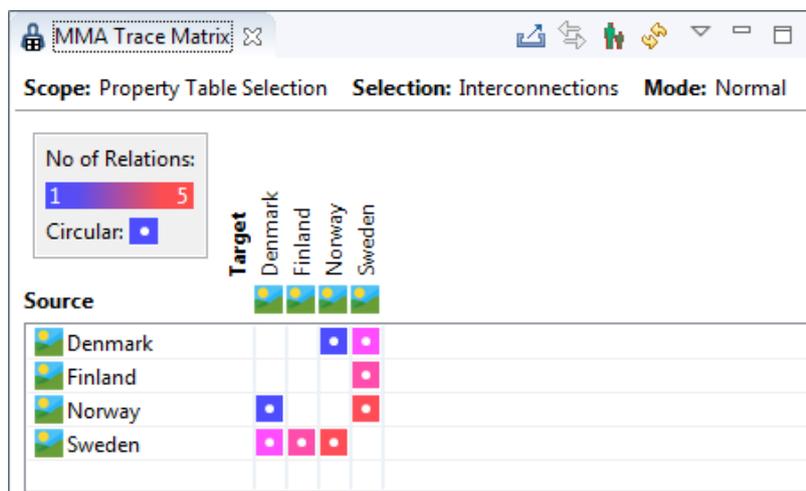
Relationship overviews

The interactive *MetaModelAgent Trace Matrix View* lets you display a variety of relationship-related information in a matrix-style layout. The view can display direct relationships as well as composite relationships, including state machine transitions and activity flows. Redundancies as well as circular relationship chains will be highlighted in this view.



MetaModelAgent Trace Matrix View

In an architectural context, the Trace Matrix View will make it easy to see which architectural elements that are related directly or indirectly to each other. For indirect relationships, the total distance will be indicated by color. As the figure below shows, it can also be used on an aggregated level, where a filled cell indicates that there are relationships between underlying elements on a lower level.



MetaModelAgent Trace Matrix View (Transitive mode)

Traceability chains

The interactive *MetaModelAgent Trace Tree View* let you display relationship chains by following a specific kind or relation or all kind of relations between selected source and target element, either forward or backward. This view will also display state machine transition paths and activity flow paths.

Element	Reference
▲ Copenhagen	
▲ Malmö	Multiple Relations
▲ Stockholm	Multiple Relations
▲ Helsinki	Baltic Sea (East)
▲ Vaasa	Multiple Relations
▲ Turku	South Kvarken (East)
▲ Vaasa	E8 (North)
▲ Oslo	Skagerrak/Kattegat (North)
▲ Stockholm	Multiple Relations
▲ Helsinki	Baltic Sea (East)
▲ Vaasa	Multiple Relations
▲ Turku	South Kvarken (East)
▲ Vaasa	E8 (North)

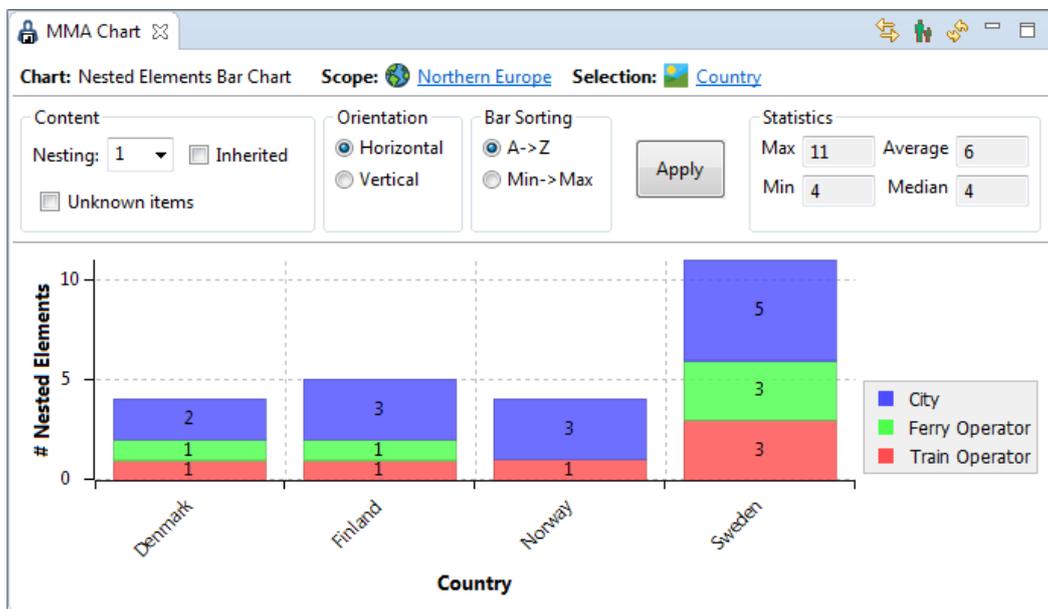
MetaModelAgent Trace Tree View

In an architectural context the Trace Tree View can give you the long-awaited traceability viewpoint crossing all architectural layers and tiers.

Dashboards

The interactive *MetaModelAgent Chart View* can display a variety of information about nested and related elements as well as distribution of property values for a specific kind of items.

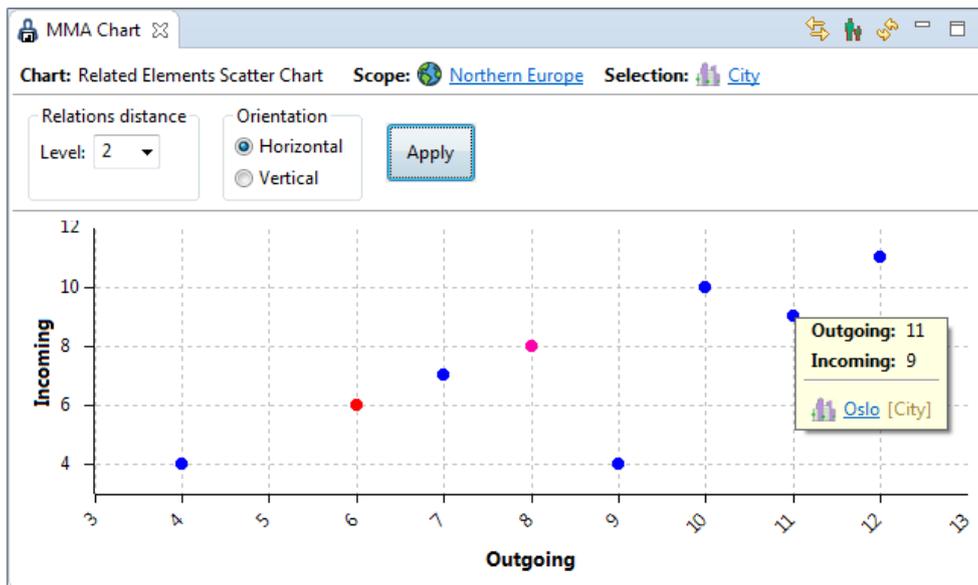
The Bar chart below displays number of directly nested elements (cities and transport operators) for each country.



MetaModelAgent Bar Chart View

In an architectural context the Chart View can, among others, reveal the largest architectural elements, and those that has the highest number of dependent elements.

The scatter chart below displays an overview of cities complexity in terms of number of incoming and outgoing connections.



MetaModelAgent Scatter Chart View

The Chart View can be used as a dashboard providing management views and executive summaries of the architecture being modeled.

Try it Yourself!

You are only a few steps away from exploring all the capabilities in Papyrus and MetaModelAgent in your own MS Windows, Linux or MacOS environment.

1. Download and install **Eclipse Papyrus** using the instructions at www.metamodelagent.com/papyrus_installation.html.
2. Download and install **Adocus MetaModelAgent** to your running Eclipse Papyrus environment using the instructions at: www.metamodelagent.com/get_started.html, where you also will find instructions on how to obtain a limited free license or a time-limited evaluation license.
3. Download and install the **Cities and Connections example** used in this paper, using the instructions at www.metamodelagent.com/download.html#demos.

For more information about Papyrus and MetaModelAgent, or if you have any question or comments about this paper, please send a mail to thomas.wiman@adocus.com.

References

- [1] Open-source Eclipse Papyrus
<https://www.eclipse.org/papyrus>
- [2] Adocus MetaModelAgent
<http://www.metamodelagent.com>
- [3] IBM Rational Software Architect Designer
<https://www.ibm.com/us-en/marketplace/rational-software-architect-designer>
- [4] Papyrus User Guide
https://wiki.eclipse.org/Papyrus_User_Guide
- [5] About UML-profiling in Papyrus
https://www.eclipse.org/papyrus/resources/PapyrusUserGuideSeries_AboutUMLProfile_v1.0.0_d20120606.pdf
- [6] CSS-Stylesheets for Papyrus diagrams
<https://wiki.eclipse.org/MDT/Papyrus/UserGuide/CSS>
- [7] MetaModelAgent Metamodeling
http://www.metamodelagent.com/documentation/MetaModelAgent_MetaModeling.pdf
- [8] Eclipse IDE Plugin Development
<https://www.vogella.com/tutorials/EclipsePlugin/article.html>
- [9] MetaModelAgent Modeling User Manual
http://www.metamodelagent.com/documentation/MetaModelAgent_UserManual.pdf
- [10] MetaModelAgent Model Analysis Manual
http://www.metamodelagent.com/documentation/MetaModelAgent_ModelAnalysis.pdf
- [11] The complete “Cities and Connection” example (zipped Eclipse project)
http://www.metamodelagent.com/demos/cities_and_connections_dsml_demo.zip